

## Information technology - SCSI Primary Commands - 4 (SPC-4)

This is an internal working document of T10, a Technical Committee of Accredited Standards Committee INCITS (InterNational Committee for Information Technology Standards). As such this is not a completed standard and has not been approved. The contents may be modified by the T10 Technical Committee. The contents are actively being modified by T10. This document is made available for review and comment only.

Permission is granted to members of INCITS, its technical committees, and their associated task groups to reproduce this document for the purposes of INCITS standardization activities without further permission, provided this notice is included. All other rights are reserved. Any duplication of this document for commercial or for-profit use is strictly prohibited.

T10 Technical Editor:

Ralph O. Weber  
ENDL Texas  
18484 Preston Road  
Suite 102 PMB 178  
Dallas, TX 75252  
USA

Telephone: 214-912-1373  
Facsimile: 972-596-2775  
Email: ROWeber at IEEE dot org

---

Reference number  
ISO/IEC 14776-454 : 200x  
ANSI INCITS.\*\*\*:200x

## **Points of Contact:**

### **T10 Chair**

John B. Lohmeyer  
LSI Corp.  
4420 Arrows West Drive  
Colorado Springs, CO 80907-3444  
Tel: (719) 533-7560  
Fax: (719) 533-7183  
Email: lohmeier at t10 dot org

### **T10 Vice-Chair**

Mark Evans  
Western Digital  
5863 Rue Ferrari  
San Jose, CA 95138  
Tel: (408) 363-5257  
Fax: (408) 363-5139  
Email: Mark dot Evans at wdc dot com

### **INCITS Secretariat**

INCITS Secretariat  
1250 Eye Street, NW Suite 200  
Washington, DC 20005  
Telephone: 202-737-8888  
Facsimile: 202-638-4922  
Email: incits@itic.org

**T10 Web Site** <http://www.t10.org>

**T10 Reflector** To subscribe send e-mail to majordomo@T10.org with 'subscribe' in message body  
To unsubscribe send e-mail to majordomo@T10.org with 'unsubscribe' in message body  
Internet address for distribution via T10 reflector: T10@T10.org

### **Document Distribution**

INCITS Online Store  
managed by Techstreet  
1327 Jones Drive  
Ann Arbor, MI 48105  
<http://www.techstreet.com/incits.html>  
Telephone: 1-734-302-7801 or  
1-800-699-9277  
Facsimile: 1-734-302-7811

or

Global Engineering  
15 Inverness Way East  
Englewood, CO 80112-5704  
<http://global.ihs.com/>  
Telephone: 1-303-792-2181 or  
1-800-854-7179  
Facsimile: 1-303-792-2192

## Revision Information

Changes in the SCSI standards family list, clause 1, are never marked with change bars. Changes in the ASC/ASCQ, Operation Code, Log Page Code, Mode Page Code and Vendor ID tables are usually not marked with change bars.

## 1 Approved Documents Included

No T10 approved proposals are included in SPC-4 r0. SPC-4 r0 is identical to SPC-3 r23 except that editorial changes made by the ANSI Editor as part of the INCITS Public Review for SPC-3 appear in SPC-4 r0.

The following T10 approved proposals have been incorporated SPC-4 up to and including this revision:

04-198r5	Background Medium Scan
04-262r8	ATA Command Pass-Through
04-335r2	WORM Tamper Read Enable [additional sense code]
04-371r2	Enable Background Operation Error Reporting Bit
04-372r1	I_T NEXUS RESET task management function
04-397r3	SAT ATA control mode page proposal
05-003r3	VPD Page 83 Identifier s/b Designator Changes
05-015r2	Add new ADC-2 command to set Volume Identifier
05-026r2	SMC-3: DTE Prevented Medium Removal [additional sense code]
05-074r1	SPC-4 Command requirements
05-142r4	SAT - LOG SENSE command and SMART [additional sense code]
05-156r7	Application ownership of protection information Reference Tag
05-157r9	Security Commands proposal
05-230r0	Obsolete Per Initiator Port in SPC-4
05-232r2	Log Command Corrections
05-242r2	Combinations of bits and fields in the LOG SELECT CDB and log parameters
05-243r4	SMC-3 REQUEST DATA TRANSFER ELEMENT INQUIRY command {new ASC only}
05-248r4	Statistics and Performance Log Pages
05-252r2	Create well known LUN for trusted commands
05-262r3	SSC-3: ASC/ASCQ for Medium Thread Failure [additional sense code]
05-284r4	Self Describing Command Timeouts
05-296r0	Persistent Reservations PREEMPT AND ABORT Clarification
05-299r1	Correct Log Page Format Tables in SPC-4, SBC-3, & SAS-2
05-317r3	Remove Attached Media Changer model
05-331r1	Allow Mode Sense Through Read Only Persistent Reservation
05-337	(Sep. 2005 CAP minutes) Change IEEE 1212-1994 reference to IEEE 1212-2001
05-340r3	Background scan additions
05-360r2	Extended self test completion time value update
05-368r2	Allow more commands through Write Exclusive reservations
05-370r2	NOTIFY (POWER LOSS EXPECTED) Fixes [additional sense code]
05-374r2	Disabling Reassign on Write Long Logical Blocks
05-375r1	EBACKERR bit Error Reporting Clarification
05-379r0	Unit attention condition trivial wording changes
05-380r0	Mode Page Policy MLUS bit correction
05-383r4	Deferred Microcode Downloads
05-406r1	New ASC/ASCQs for Device Port Address
06-018r0	Update SPC-4 version field
06-079r1	More Log Page Cleanup
06-108r3	Add security protocol and additional sense for tape encryption
06-113r2	Log Parameter Subpages

- 06-127 Assign Security Protocol code value EFh to the ATA Device Server Password Security protocol (minutes of the March Plenary)
- 06-147r0 ALL\_TG\_PT clarification
- 06-175r2 Wrapping and saturating counter definitions
- 06-191r0 Request for the reservation/assignment of Security Protocol value
- 06-195r1 Persistent Reservation additional sense code changes
- 06-196r0 Persistent Reservation REGISTER AND MOVE from unregistered I\_T nexus
- 06-198r0 Making the basic task management model obsolete
- 06-219r1 New ASC/ASCQs for LOGICAL UNIT ACCESS NOT AUTHORIZED
- 06-221r3 Peripheral device identifying information
- 06-223r1 Vendor-specific service actions for MAINTENANCE (IN) and MAINTENANCE (OUT)
- 06-224r1 Change 'medium changer' to 'media changer'
- 06-248r1 Remove the PREVENT ALLOW MEDIUM REMOVAL command from SPC-4
- 06-259r1 Making linked commands obsolete
- 06-264r2 REQUEST SENSE parameter data clarifications
- 06-274r1 REQUEST SENSE and Stopped power condition
- 06-278r0 Fix persistence of SET DEVICE IDENTIFIER
- 06-282r4 SPC-4 WRITE BUFFER clarifications
- 06-298r0 Persistent Reservations correction
- 06-320r1 Statistics logging for FUA and FUA\_NV
- 06-321r1 Statistics and Performance log pages command cleanup
- 06-323r1 Multiple service delivery subsystem editorial tweaks
- 06-325r0 New ASC/ASCQ for SAT queue cleared condition
- 06-337r0 Statistics Group Number inconsistencies in SPC-4, SSC-3, and MMC-5
- 06-350r0 Power conditions state machine clarifications
- 06-362r7 Error History proposal
- 06-369r8 Security Association Model for SPC-4
- 06-388r3 Security Goals and Threat Model
- 06-389r5 Using Public-Key Cryptography for Key Wrapping {new ASC only}
- 06-390r2 ALUA/TPGS Disconnected state
- 06-411r2 Clear REPORTED LUNS DATA HAS CHANGED on any command
- 06-454r0 Clarifying Identifying Information Types Requirements
- 06-455r1 Lost SET IDENTIFYING INFORMATION Parameter List Length requirement
- 06-465r1 Alternative Proposal for Management Transport
- 06-484r1 Encryption MAM Attribute
- 07-005r2 SSC-3: Encryption unsupported medium {new ASC only}
- 07-008r8 Expander Notification of Temporary Shutdown
- 07-022r1 MANAGEMENT PROTOCOL IN command field name fixup
- 07-029r3 Extended SCSI Commands
- 07-056r0 Editorial change: IEEE P1667 to IEEE 1667
- 07-077r0 Stop fruitless attempts to shackle users via the IALUAE Control Ext. Mode bit
- 07-080r2 Persistent reservations self-preempting clarification
- 07-100r0 Digital Signing of Microcode
- 07-111r1 Duplicate persistent reservation wording removal
- 07-153r1 Protocol-Specific VPD pages
- 07-159r1 Add QTSS and QUAS bits to REPORT SUPPORTED TMF
- 07-169r7 ESP-SCSI for Parameter Data
- 07-172r1 Target Port Group membership clarification
- 07-184r0 Request for security protocol
- 07-196r0 LOG SELECT CDB page/subpage field usage
- 07-203r0 Block Device Characteristics VPD page and medium rotation rate field
- 07-215r0 Protocol-Specific log page subpages
- 07-216r1 NAA Locally Assigned designator format
- 07-218r3 SSC-3: Configurable Early Warning {new ASC only}

07-222r2 SMC-3 Additional sense code for incompatible destination element {new ASC only}  
 07-244r3 SMC-3 READ ATTRIBUTE and WRITE ATTRIBUTE command clarification {new ASC only}  
 07-257r1 Prohibited needed as a keyword in SPC-4  
 07-262r6 Command Security Model [Weber/Penokie]  
 07-269r1 Update Annex C for ADC-2 changes  
 07-281r1 "Except INQUIRY, REPORT LUNS, and REQUEST SENSE"  
 07-295r1 Request for TrustedFlash Security Protocol ID Value  
 07-315 SECURITY CONFLICT IN TRANSLATED DEVICE additional sense code for block devices {new ASC only} (minutes of September CAP WG)  
 07-361r6 SSC-3 Out of Band Encryption Key Management {new ASCs only}  
 07-389r1 Persistent Reservation Clarification  
 07-390r1 Target Port Groups & Logical Unit Groups  
 07-430r2 UML model for CbCS  
 07-437r4 Establishing a Security Association using IKEv2  
 07-451r1 WRITE LONG COR\_DIS and WR\_UNCOR interaction  
 07-459r4 Unit attention condition queuing  
 08-008r2 Download microcode I\_T nexus usage  
 08-029r2 Automation Encryption Control {new ASCs only}  
 08-034r2 Error history cleanup  
 08-037r3 SA Creation corrections and clarifications  
 08-041r1 Use period as decimal separator in T10 standards  
 08-053r1 Persistent reservation fixes  
 08-057r0 Allocation length advisories in Inquiry command  
 08-061 ATA Power Conditions mode subpage assignment (January CAP Minutes)  
 08-108r1 Completing the SET IDENTIFYING INFORMATION command  
 08-113r2 UML Conventions and CbCS UML Updates  
 08-116r3 Protection Type 3 Reference Tag Clarification [Penokie]  
 08-138r1 Constraints on SPC-4 SA creation based on Usage Type  
 08-145r3 Capability-based Command Security (CbCS) [the rewrite] [Weber]  
 08-156r2 Non-volatile cache becoming volatile [Penokie]  
 08-158r1 ESP-SCSI field alignments  
 08-177r0 Bogus Sense Key in SA Creation Parameter Data Error Handling [Weber]  
 08-178r0 IKEv2-SCSI Message ID field size correction [Weber]  
 08-180r0 SCSI device name cleanup in SPC-4 [Weber]  
 08-194r1 Common definition for CODE SET [Ballard]  
 08-199r1 Even more log page cleanup [Evans]  
 08-214r1 Allowing ESP-SCSI descriptors in variable length CDBs [Weber]  
 08-231r1 SPEC\_I\_PT clarification [Knight]  
 08-254r0 Protocol Specific diagnostic page [Elliott]  
 08-261r1 Parsing variable-length parameter lists and data [Elliott]  
 08-262r0 SAI, SQN, IV, & Data must be contiguous [Weber]  
 08-287r1 ATA Information VPD page nits [Elliott]

The following proposal was approved by T10 but not included in this revision because numerous and complex corrections were required to align it with SA creation and other relatively new SPC features:

T10 approved 07-027r2 (Enabling and disabling Transport Layer Retries), r1 as revised, for inclusion in SPC-4 but the author decided not to include SPC-4 changes in the final revision. Instead, a new proposal 07-153 was started to cover the same information.

T10 approved 07-454r5 (Capability based Command Security), but the proposal contained a large number of problems that required substantive corrections. Later, 08-145r3 (Capability-based Command Security (CbCS) [the rewrite]) was approved as the replacement for 07-454r5 and duly incorporated as noted above.

To the best of the technical editor's knowledge, all T10 approved proposals have been included in this revision.

## 2 Revision History

### 2.1 Revision 0 (1 August 2005)

Revision 0 of SPC-4 is substantially equal to revision 23 of SPC-3. The only differences arise from changes made in SPC-3 by the ANSI Editor during the INCITS Public Review process.

### 2.2 Revision 1 (3 September 2005)

The following T10 approved proposals were incorporated in SPC-4 revision 1:

04-198r5	Background Medium Scan
04-262r8	ATA Command Pass-Through
04-371r2	Enable Background Operation Error Reporting Bit
04-372r1	I_T NEXUS RESET task management function
04-397r3	SAT ATA control mode page proposal
05-003r3	VPD Page 83 Identifier s/b Designator Changes
05-015r2	Add new ADC-2 command to set Volume Identifier
05-074r1	SPC-4 Command requirements
05-230r0	Obsolete Per Initiator Port in SPC-4

The sentence introducing the 'Parameter control bits for Informational exceptions log parameter' table was changed from 'self test results log parameters' to 'informational exceptions log parameter'.

Media Changer usage was added for the LOGICAL UNIT NOT READY, OFFLINE additional sense code.

A missing column separator line was added in Table C.5 (LOG SELECT save options).

Several purely editorial changes made by the ANSI Editor in the published version of SPC-3 were included in SPC-4 r01.

### 2.3 Revision 1a (5 September 2005)

The ASCQ assignment for ATA DEVICE FAILED SET FEATURES was changed from F1h to 71h so that it did not fall in the vendor specific ASCQ range.

### 2.4 Revision 2 (15 September 2005)

The following T10 approved proposals were incorporated in this SPC-4 revision:

05-296r0	Persistent Reservations PREEMPT AND ABORT Clarification
05-299r1	Correct Log Page Format Tables in SPC-4, SBC-3, & SAS-2
05-331r1	Allow Mode Sense Through Read Only Persistent Reservation
05-337	(Sep. 2005 CAP minutes) Change IEEE 1212-1994 reference to IEEE 1212-2001

For consistency with other proposals, one instance of 'background medium read scan' was changed to 'background medium scan'.

The page codes for the Background Control mode page and Background Scan Results log page were added to the Numeric Order Codes annex.

## 2.5 Revision 3 (19 January 2006)

The following T10 approved proposals were incorporated in this SPC-4 revision:

04-335r2	WORM Tamper Read Enable [additional sense code]
05-026r2	SMC-3: DTE Prevented Medium Removal [additional sense code]
05-142r4	SAT - LOG SENSE command and SMART [additional sense code]
05-156r7	Application ownership of protection information Reference Tag
05-157r9	Security Commands proposal
05-242r2	Combinations of bits and fields in the LOG SELECT CDB and log parameters
05-252r2	Create well known LUN for trusted commands
05-262r3	SSC-3: ASC/ASCQ for Medium Thread Failure [additional sense code]
05-317r3	Remove Attached Media Changer model
05-340r3	Background scan additions
05-360r2	Extended self test completion time value update
05-368r2	Allow more commands through Write Exclusive reservations
05-370r2	NOTIFY (POWER LOSS EXPECTED) Fixes [additional sense code]
05-374r2	Disabling Reassign on Write Long Logical Blocks
05-375r1	EBACKERR bit Error Reporting Clarification
05-379r0	Unit attention condition trivial wording changes
05-380r0	Mode Page Policy MLUS bit correction
05-383r4	Deferred Microcode Downloads
05-406r1	New ASC/ASCQs for Device Port Address
06-018r0	Update SPC-4 version field

In the MAM device type attributes table, code value 020Ah was removed from the reserved list because the same table show it as allocated to DEVICE MAKE/SERIAL NUMBER AT LAST LOAD.

In the mode page codes and subpage codes table, the subpage code value for page code 00h was changed from 'not applicable' to 'vendor specific'. Since mode page code 00h is vendor specific, the applicability of subpage codes must also be vendor specific.

The mode page code information in the numeric order codes informative annex was changed to match the normative code assignments in Mode Parameters subclause. Every code described in the annex as vendor specific was changed to device-type specific, with the exception of page code 00h, which the normative text defines as vendor specific.

Several additional sense code definitions were added as requested by the MMC-5 working group.

Reference columns were added to the mode tables for READ BUFFER and WRITE BUFFER. Several instances of 'accept' were changed to 'process' as in 'If the device server is unable to ~~accept~~ process ...'

Some (but not all) of the editorial changes in 05-340r2 that were removed by the CAP working group in r3 were made anyway.

## 2.6 Revision 4 (9 March 2006)

The following T10 approved proposals were incorporated in this SPC-4 revision:

- 06-079r1 More Log Page Cleanup
- 06-127 Assign Security Protocol code value EFh to the ATA Device Server Password Security protocol (minutes of the March Plenary)
- 06-147r0 ALL\_TG\_PT clarification

The descriptions of three additional sense codes 10h/01h, 10h/02h, and 10h/03h were modified to match the nomenclature in SBC-2 and SBC-3 as follows:

- a) ~~DATA~~ LOGICAL BLOCK GUARD CHECK FAILED;
- b) ~~DATA~~ LOGICAL BLOCK APPLICATION TAG CHECK FAILED; and
- c) ~~DATA~~ LOGICAL BLOCK REFERENCE TAG CHECK FAILED

The Operation Codes table in the numerical codes annex was updated to:

- a) Make the SECURITY PROTOCOL IN and SECURITY PROTOCOL OUT commands optional for MMC devices; and
- b) Add the OPEN/CLOSE IMPORT/EXPORT ELEMENT command for SMC devices.

The ADC usage of Service Buffers Information log page was added to the numerical codes annex.

The supported protection type (SPT) field in the Extended INQUIRY Data VPD page was updated to match the SBC-3 usage.

The definition of additional sense code was update to match existing field reference nomenclature.

## 2.7 Revision 5 (10 June 2006)

The following T10 approved proposals were incorporated in this SPC-4 revision:

- 05-248r4 Statistics and Performance Log Pages
- 06-108r3 Add security protocol and additional sense for tape encryption
- 06-113r2 Log Parameter Subpages
- 06-191r0 Request for the reservation/assignment of Security Protocol value
- 06-195r1 Persistent Reservation additional sense code changes
- 06-196r0 Persistent Reservation REGISTER AND MOVE from unregistered I\_T nexus
- 06-198r0 Making the basic task management model obsolete
- 06-219r1 New ASC/ASCQs for LOGICAL UNIT ACCESS NOT AUTHORIZED
- 06-223r1 Vendor-specific service actions for MAINTENANCE (IN) and MAINTENANCE (OUT)
- 06-224r1 Change 'medium changer' to 'media changer'
- 06-248r1 Remove the PREVENT ALLOW MEDIUM REMOVAL command from SPC-4

The operation code list in the informative annex was updated to match MMC-5 usage. The log page code list was updated to match SSC-3.

A reference to the SSCS bit was corrected to the SCCS bit.

The References Under Development subclause was updated.

The MAM device attributes summary table was updated to match the text.



The annex describing Persistent Reservations functionality needed for an implementation equivalent to the model for Reserve/Release was corrected to note that the PERSISTENT RESERVE IN command has no equivalent in the Reserve/Release model.

Numerous editorial changes were made as suggested in 06-271r0. Other minor editorial corrections were made as per requests from T10 members received privately and on the T10 reflector.

## 2.8 Revision 5a (14 June 2006)

Fix errors, both significant and trivial, in r05. Change bars are cumulative for r05 and r05a.

The revision number in 2.7 was corrected.

Log page parameter code nomenclature was made consistent in the various clause 7 format tables.

The Time Interval log parameter was corrected from 0002h to 0003h. Clarifying changes were made to adjust for moving the *command weight*, *weighted command time*, and *idle time* definitions from the Statistics and Performance log pages overview to the Statistics and Performance log parameter definition. The handling of task priority values of zero was returned to what was defined in 05-248r4. The removal of the TIME INTERVAL DESCRIPTOR field was reflected throughout. The READ/WRITE COMMAND PROCESSING INTERVAL field references were corrected to READ/WRITE COMMAND PROCESSING TIME field.

The coded value tables headers were updated to reference SBC-3, SSC-3, MMC-6, and SMC-3. Hopefully, these will be the active standards at the time SPC-4 is published.

## 2.9 Revision 6 (18 July 2006)

The following T10 approved proposals were incorporated in this SPC-4 revision:

- 06-175r2 Wrapping and saturating counter definitions
- 06-259r1 Making linked commands obsolete
- 06-264r2 REQUEST SENSE parameter data clarifications
- 06-274r1 REQUEST SENSE and Stopped power condition
- 06-278r0 Fix persistence of SET DEVICE IDENTIFIER
- 06-298r0 Persistent Reservations correction
- 06-320r1 Statistics logging for FUA and FUA\_NV
- 06-323r1 Multiple service delivery subsystem editorial tweaks
- 06-325r0 New ASC/ASCQ for SAT queue cleared condition
- 06-337r0 Statistics Group Number inconsistencies in SPC-4, SSC-3, and MMC-5

In the table showing the format of the Statistics and Performance log parameter, the field names were corrected to match the field names used in the field definitions that follow the table.

Following a suggestion indicated by 06-323r1, all practical instances of SAM-3 were changed to SAM-4.

## 2.10 Revision 7 (6 October 2006)

The following T10 approved proposals were incorporated in this SPC-4 revision:

- 06-221r3 Peripheral device identifying information
- 06-350r0 Power conditions state machine clarifications

The SECURITY PROTOCOL SPECIFIC field for certificates in the SECURITY PROTOCOL IN command was corrected to 0001h. The erroneous value in previous revisions was 0000h. The Certificate data table title and the cross reference to the table were corrected in the same subclause.

In the SECURITY PROTOCOL OUT CDB format table, ALLOCATION LENGTH was changed to TRANSFER LENGTH to match the field definition text following the table.

## 2.11 Revision 7a (7 October 2006)

Revision 7a contains the same content as revision 7. The source files have been converted to FrameMaker 7.2 and tables like the additional sense code table have been restructured to use a single column for the DTL... information. The latter change reduces the complexity (i.e., size) of the FrameMaker source files substantially and (hopefully) eliminates a bazillion error messages from the Acrobat PDF Distiller.

## 2.12 Revision 8 (6 January 2007)

The following T10 approved proposals were incorporated in this SPC-4 revision:

- 05-232r2 Log Command Corrections
- 05-284r4 Self Describing Command Timeouts
- 06-282r4 SPC-4 WRITE BUFFER clarifications
- 06-321r1 Statistics and Performance log pages command cleanup
- 06-388r3 Security Goals and Threat Model
- 06-411r2 Clear REPORTED LUNS DATA HAS CHANGED on any command
- 06-454r0 Clarifying Identifying Information Types Requirements
- 06-455r1 Lost SET IDENTIFYING INFORMATION Parameter List Length requirement
- 06-465r1 Alternative Proposal for Management Transport
- 06-484r1 Encryption MAM Attribute

The CDB formats for the LOG SENSE and LOG SELECT commands were corrected to be 10 bytes in length.

The cross reference for the IEEE Extended description was corrected.

The information about Fibre Channel standards used by SCSI devices was substantially updated.

The codes annex description for log page 11h was updated.

The contact information for ordering ANSI standards was updated as per a request received from the ANSI editor.

A few less significant wording errors were corrected.

## 2.13 Revision 9 (17 February 2007)

The following T10 approved proposals were incorporated in this SPC-4 revision:

- 06-390r2 ALUA/TPGS Disconnected state
- 07-022r1 MANAGEMENT PROTOCOL IN command field name fixup

The editing pencil was applied liberally to 06-390r2 during its incorporation. The most heavy-handed changes were in the glossary, where 'primary' and 'secondary' versions of asymmetric access terms were added beyond those shown in the proposal. Where necessary, the glossary changes were reflected throughout the working draft. The

'Introduction to asymmetric logical unit access' subclause was heavily restructured to clarify the secondary target port group rules agreed upon by the CAP working group.

An error made during the incorporation of 05-242r2 in r03 which affected two rows in the table titled 'PCR bit, SP bit, and PC field meanings when parameter list length is zero' in the LOG SELECT command definition was corrected.

In the REPORT SUPPORTED OPERATION CODES command, the example CDB usage bit map was updated from 'A3h, 0Ch, 03h, FFh, FFh, FFh, FFh, FFh, FFh, FFh, 00h, 07h' to 'A3h, 0Ch, 87h, FFh, FFh, FFh, FFh, FFh, FFh, FFh, 00h, 07h' in order to reflect recent changes in the format of the REPORT SUPPORTED OPERATION CODES CDB.

The ATA Information VPD page was added to the appropriate tables with references to SAT-2 where needed.

The Normative References and References Under Development were updated to reflect the publication of some projects and the initiation of new ones.

## 2.14 Revision 10 (21 April 2007)

The following T10 approved proposals were incorporated in this SPC-4 revision:

- 06-369r8 Security Association Model for SPC-4
- 06-389r5 Using Public-Key Cryptography for Key Wrapping {new ASC only}
- 07-005r2 SSC-3: Encryption unsupported medium {new ASC only}
- 07-008r8 Expander Notification of Temporary Shutdown
- 07-056r0 Editorial change: IEEE P1667 to IEEE 1667
- 07-077r0 Stop fruitless attempts to shackle users via the IALUAE Control Ext. Mode bit
- 07-080r2 Persistent reservations self-preempting clarification
- 07-100r0 Digital Signing of Microcode
- 07-111r1 Duplicate persistent reservation wording removal

In the LOG SELECT command, a brief explanation was added to clarify that parameter data contains log page codes and subpage codes and this makes the use of the CDB fields illegal when parameter data is present.

The larger page length in the General Statistics and Performance log page table was corrected to match the length shown at the bottom of the table.

Several incorrect uses of 'target port group asymmetric access state' were corrected to 'target port asymmetric access state'.

In the numeric codes informative annex, T (i.e., SSC-x tapes support) was added Device Statistics log page.

In tables listing mode pages, the Protocol Specific LUN name was corrected to Protocol Specific Logical Unit.

In the Conventions subclause, the unclear word 'comprised' was globally replaced with 'consisting'.

Several problems with small-caps field names (e.g., field names not followed by the word field) were corrected. MSB and LSB were added to the BUFFER CAPACITY field in the READ BUFFER descriptor. Several other editorial mistakes were corrected.

## 2.15 Revision 11 (14 May 2007)

The following T10 approved proposals were incorporated in this SPC-4 revision:

- 07-159r1 Add QTSS and QUAS bits to REPORT SUPPORTED TMF
- 07-172r1 Target Port Group membership clarification
- 07-184r0 Request for security protocol
- 07-196r0 LOG SELECT CDB page/subpage field usage
- 07-203r0 Block Device Characteristics VPD page and medium rotation rate field
- 07-216r1 NAA Locally Assigned designator format

In the introductory sentence for the Mandatory/Optional access controls resources tables, 'persistent' was corrected to 'persist'.

## 2.16 Revision 12 (11 January 2008)

The following T10 approved proposals were incorporated in this SPC-4 revision:

- 06-362r7 Error History proposal
- 07-029r3 Extended SCSI Commands
- 07-153r1 Protocol-Specific VPD pages
- 07-215r0 Protocol-Specific log page subpages
- 07-257r1 Prohibited needed as a keyword in SPC-4
- 07-262r6 Command Security Model [Weber/Penokie]
- 07-269r1 Update Annex C for ADC-2 changes
- 07-281r1 "Except INQUIRY, REPORT LUNS, and REQUEST SENSE"
- 07-389r1 Persistent Reservation Clarification
- 07-390r1 Target Port Groups & Logical Unit Groups
- 07-430r2 UML model for CbCS
- 07-437r4 Establishing a Security Association using IKEv2

Additional sense codes defined in following T10 approved proposals were incorporated in this SPC-4 revision:

- 07-218r3 SSC-3: Configurable Early Warning
- 07-222r2 SMC-3 Additional sense code for incompatible destination element
- 07-244r3 SMC-3 READ ATTRIBUTE and WRITE ATTRIBUTE command clarification
- 07-315 SECURITY CONFLICT IN TRANSLATED DEVICE additional sense code for block devices (minutes of September CAP WG)

The following proposal was approved by T10 but not included in this revision because the referenced web site contained no information about standards development at the time this revision was produced:

- 07-295r1 Request for TrustedFlash Security Protocol ID Value

The correct year was added to the SBP-3 normative reference.

The UCS normative reference was updated to ISO/IEC 10646:2003/Amd.2:2006.

The codes annex was modified to indicate VPD page support by RBC devices and to add several device-type specific VPD pages.

In the PREEMPT service action flowchart figure, three instances of 'registration' (singular) were changed to 'registrations' (plural). These changes are not marked with change bars.

The first paragraph of the Informational Exceptions Control mode page was updated to use 'e.g.' language that modernizes its discussion of the ADDITIONAL SENSE CODE field.

In the security model, several instances of DC\_xxx SA parameters were changed to the correct DS\_xxx (device server).

The Extended INQUIRY Data VPD page definition was edited heavily to make the nomenclature consistent for all field definitions.

A consistent nomenclature of 'blah blah descriptor [first]' was applied to all table formats that contain first/last descriptors.

Several of 'generate unit attention ...' were changed to use 'establish'.

Several instances of 'specifies' were changed to 'indicates' in the VPD page definitions. VPD pages cannot specify anything because they are read only.

The mode page codes table nomenclature for other SCSI standards was updated to match the glossary.

A table footer was added to reference the numeric codes annex in the diagnostic page codes table.

The codes annex diagnostic pages table was updated to match SES-2.

The codes annex log page table was modified to match the normative text.

Several typographical errors in the READ BUFFER command were corrected.

Other obvious typos were corrected and not marked with change bars.

New codes annex tables were added for transport protocol specific log, mode, and VPD pages and VPD pages.

## 2.17 Revision 13 (11 March 2008)

The following T10 approved proposals were incorporated in this SPC-4 revision:

05-243r4	SMC-3 REQUEST DATA TRANSFER ELEMENT INQUIRY command {new ASC only}
07-169r7	ESP-SCSI for Parameter Data
07-295r1	Request for TrustedFlash Security Protocol ID Value
07-451r1	WRITE LONG COR_DIS and WR_UNCOR interaction
07-459r4	Unit attention condition queuing
08-034r2	Error history cleanup
08-037r3	SA Creation corrections and clarifications
08-041r1	Use period as decimal separator in T10 standards
08-053r1	Persistent reservation fixes
08-057r0	Allocation length advisories in Inquiry command
08-061	ATA Power Conditions mode subpage assignment (January CAP Minutes)

Additional sense codes defined in following T10 approved proposals were incorporated in this SPC-4 revision:

07-361r6	SSC-3 Out of Band Encryption Key Management
08-029r2	Automation Encryption Control

Incorporation of 07-295r1 was accomplished by changing the reference to the Secure Digital Card Association and their web site where the TrustedFlash Security Protocol specification is published.

To maintain consistent terminology with SAM-4, several instances of 'task priority' were changed to 'command priority'.

In the Extended INQUIRY Data VPD page definition, a cross reference was added to define the location of the PROTECT bit, which happens to be located in the Standard INQUIRY Data.

In the fixed format sense data definition, the descriptions for the ILI bit, FILEMARK bit, and EOM bit were modified to align them with the description of the INFORMATION field. The concepts found in the r12 descriptions of these bits were turned into 'e.g.' examples.

Two confusing instances of 'single algorithm' were changed to 'combined integrity check and encryption algorithm' in the IKEv2-SCSI Authentication step model subclause. Wording problems related to combined modes were fixed in protocol details informative annex.

The reference for the ATA Device Server Password Security security protocol was changed to SAT-2.

Since SAT-2 does not support tape devices, the T was removed from the codes annex rows for the Parallel ATA Control and Serial ATA Control mode pages.

A UML reference was added to clause 2. UML notation conventions based on the SAM-4 conventions were added to clause 3.

The reference on the restricted bytes in the REPORT DEVICE IDENTIFYING INFORMATION command table was corrected to read SCC-2.

A normative reference to SSC-2 was added because of the reference to the RSMK bit, which is obsolete in SSC-3.

Other editorial corrections to minor to identify were also made.

## 2.18 Revision 14 (17 March 2008)

The following T10 approved proposals were incorporated in this SPC-4 revision:

- 08-008r2 Download microcode I\_T nexus usage
- 08-108r1 Completing the SET IDENTIFYING INFORMATION command
- 08-113r2 UML Conventions and CbCS UML Updates
- 08-138r1 Constraints on SPC-4 SA creation based on Usage Type
- 08-158r1 ESP-SCSI field alignments

Problems with SMC-2 related additional sense code definitions were corrected.

In the IKEv2-SCSI SA Cryptographic Algorithms payload format, the required value in the NUMBER OF ALGORITHM DESCRIPTORS field was corrected from five to six so that it matches the number of entries in the ordered list of algorithm descriptors.

08-008r2 failed to cover the case of non-zero buffer offsets received on the established I\_T nexus. The obvious intent was that such commands be processed as described elsewhere in the subclause. This was added as an editorial change.

Based on input from the SAM-4 letter ballot review, all instance of specified status return values were evaluated for the usage of 'terminated' or 'completed' in the description of the command. With the exception of CHECK CONDITION status, commands were described as being 'completed'. For CHECK CONDITION status, the command was described as being 'terminated'. Where such wording could not be applied the terms 'return' and 'response' were used. A few instance of 'task' were replaced with 'command'.

'Principals of SAs' was changed to 'Principles of SAs'.

## 2.19 Revision 15 (20 June 2008)

The following T10 approved proposals were incorporated in this SPC-4 revision:

08-145r3	Capability-based Command Security (CbCS) [the rewrite] [Weber]
08-156r2	Non-volatile cache becoming volatile [Penokie]
08-177r0	Bogus Sense Key in SA Creation Parameter Data Error Handling [Weber]
08-178r0	IKEv2-SCSI Message ID field size correction [Weber]
08-180r0	SCSI device name cleanup in SPC-4 [Weber]
08-194r1	Common definition for CODE SET [Ballard]
08-199r1	Even more log page cleanup [Evans]
08-214r1	Allowing ESP-SCSI descriptors in variable length CDBs [Weber]
08-231r1	SPEC_I_PT clarification [Knight]

The instruction in 08-053r1 (Persistent reservation fixes) were not fully implemented in the 'Register behaviors for a REGISTER AND IGNORE EXISTING KEY service action' table during the preparation of r13. Dangling rows were removed in this revision.

During the incorporation of 08-214r1, several instances where SA protection was restricted to command parameter data were broadened to cover selected CDBs too.

To match SAM-4 Letter Ballot changes:

- a) QUERY UNIT ATTENTION was changed to QUERY ASYNCHRONOUS EVENT in the REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS command definition; and
- b) 'initial priority' was changed to 'initial command priority' in the REPORT PRIORITY command definition, SET PRIORITY command definition, and the Control Extension mode page.

References to the under development ADC-3 were added wherever appropriate to support the Data Encryption Configuration security protocol.

The POWER CONDITION field name was modified to match the spelling in SBC-3 throughout.

Bogus IKeV2-SCSI Timeout Values an Identification – Device Server next payload code values as well as incorrect references to the IKeV2-SCSI Timeout Values payload were corrected in the per/step next payloads table.

Several mode page entries in the numeric codes annex were updated to match the latest ADC-3 revision.

Several grammar errors were corrected and not marked with change bars.

In hopes of getting a jump on ISO SPC-4, all uses of dashes in range expressions (e.g., 5Ah-FFh) were changed to use the word 'to' (e.g., 5Ah to FFh). These changes were not marked with change bars.

## 2.20 Revision 16 (29 July 2008)

The following T10 approved proposals were incorporated in this SPC-4 revision:

08-116r3	Protection Type 3 Reference Tag Clarification [Penokie]
08-254r0	Protocol Specific diagnostic page [Elliott]
08-261r1	Parsing variable-length parameter lists and data [Elliott]
08-262r0	SAI, SQN, IV, & Data must be contiguous [Weber]
08-287r1	ATA Information VPD page nits [Elliott]

In the table defining the ID TYPE field for the Identification – Application Client payload and Identification – Device Server payload, the codepoint values for ID\_FC\_NAME and ID\_KEY\_ID were swapped so that they match the values in RFC 4306.

The conventions subclause was updated to include the latest descriptions of the conventions for number value ranges and lettered/numbered lists. Where the new number convention allowed, 'through' was replaced with 'to'.

Several incorrect subclause references discovered during the ISO SPC-3 review and an incorrect table reference were corrected but not marked with change bars.



**ANSI (r)**  
**INCITS.\*\*\*:200x**

**American National Standards  
for Information Systems -**

**SCSI Primary Commands - 4 (SPC-4)**

Secretariat  
**InterNational Committee for Information Technology Standards**

Approved mm dd yy

**American National Standards Institute, Inc.**

**Abstract**

This standard defines the device model for all SCSI devices. This standard defines the SCSI commands that are basic to every device model and the SCSI commands that may apply to any device model.

## **American National Standard**

Approval of an American National Standard requires verification by ANSI that the requirements for due process, consensus, and other criteria for approval have been met by the standards developer. Consensus is established when, in the judgment of the ANSI Board of Standards Review, substantial agreement has been reached by directly and materially affected interests. Substantial agreement means much more than a simple majority, but not necessarily unanimity. Consensus requires that all views and objections be considered and that effort be made toward their resolution.

The use of American National Standards is completely voluntary; their existence does not in any respect preclude anyone, whether he or she has approved the standards or not, from manufacturing, marketing, purchasing, or using products, processes, or procedures not confirming to the standards.

The American National Standards Institute does not develop standards and will in no circumstances give interpretation on any American National Standard in the name of the American National Standards Institute. Requests for interpretations should be addressed to the secretariat or sponsor whose name appears on the title page of this standard.

**CAUTION NOTICE:** This American National Standard may be revised or withdrawn at any time. The procedures of the American National Standards Institute require that action be taken periodically to reaffirm, revise, or withdraw this standard. Purchasers of American National Standards may receive current information on all standards by calling or writing the American National Standards Institute.

**CAUTION:** The developers of this standard have requested that holders of patents that may be required for the implementation of the standard, disclose such patents to the publisher. However, neither the developers nor the publisher have undertaken a patent search in order to identify which, if any, patents may apply to this standard.

As of the date of publication of this standard, following calls for the identification of patents that may be required for the implementation of the standard, notice of one or more claims has been received.

By publication of this standard, no position is taken with respect to the validity of this claim or of any rights in connection therewith. The known patent holder(s) has (have), however, filed a statement of willingness to grant a license under these rights on reasonable and nondiscriminatory terms and conditions to applicants desiring to obtain such a license. Details may be obtained from the publisher.

No further patent search is conducted by the developer or the publisher in respect to any standard it processes. No representation is made or implied that licenses are not required to avoid infringement in the use of this standard.

Published by  
**American National Standards Institute**  
**11 West 42nd Street, New York, NY 10036**

Copyright 7/29/08 by American National Standards  
Institute  
All rights reserved.

**Printed in the United States of America**

## Contents

	Page
Foreword .....	xlv
Introduction .....	xlvii
1 Scope .....	1
2 Normative references .....	5
2.1 General .....	5
2.2 Approved references .....	5
2.3 References under development .....	7
2.4 NIST References .....	8
2.5 IETF References .....	8
2.6 OMG references .....	9
3 Definitions, symbols, abbreviations, and conventions .....	11
3.1 Definitions .....	11
3.2 Symbols and acronyms .....	22
3.3 Keywords .....	24
3.4 Conventions .....	25
3.5 Bit and byte ordering .....	27
3.6 Notation conventions .....	27
3.6.1 Notation for byte encoded character strings .....	27
3.6.2 Notation for procedure calls .....	28
3.6.3 Notation for state diagrams .....	29
3.6.4 Notation for binary power multipliers .....	29
3.6.5 Notation for UML figures .....	30
3.6.5.1 Introduction .....	30
3.6.5.2 Class notation .....	31
3.6.5.3 Class association relationships notation .....	32
3.6.5.4 Class aggregation relationships notation .....	33
3.6.5.5 Class generalization relationships notation .....	34
3.6.5.6 Class dependency relationships notation .....	35
3.6.5.7 Object notation .....	35
4 General Concepts .....	36
4.1 Introduction .....	36
4.2 The request-response model .....	36
4.3 The Command Descriptor Block (CDB) .....	36
4.3.1 CDB usage and structure .....	36
4.3.2 The fixed length CDB formats .....	37
4.3.3 The variable length CDB formats .....	41
4.3.4 Extended CDBs .....	42
4.3.4.1 XCDB model .....	42
4.3.4.2 The XCDB format .....	43
4.3.5 Common CDB fields .....	44
4.3.5.1 Operation code .....	44
4.3.5.2 Service action .....	45
4.3.5.3 Logical block address .....	45
4.3.5.4 Transfer length .....	45
4.3.5.5 Parameter list length .....	46
4.3.5.6 Allocation length .....	46

4.3.5.7 Control .....	46
4.4 Data field requirements .....	46
4.4.1 ASCII data field requirements .....	46
4.4.2 Null data field termination and zero padding requirements .....	47
4.4.3 Variable type data field requirements .....	47
4.5 Sense data .....	47
4.5.1 Sense data introduction .....	47
4.5.2 Descriptor format sense data .....	49
4.5.2.1 Descriptor format sense data overview .....	49
4.5.2.2 Information sense data descriptor .....	51
4.5.2.3 Command-specific information sense data descriptor .....	51
4.5.2.4 Sense key specific sense data descriptor .....	52
4.5.2.4.1 Sense key specific sense data descriptor introduction .....	52
4.5.2.4.2 Field pointer sense key specific data .....	53
4.5.2.4.3 Actual retry count sense key specific data .....	54
4.5.2.4.4 Progress indication sense key specific data .....	54
4.5.2.4.5 Segment pointer sense key specific data .....	55
4.5.2.4.6 Unit attention condition queue overflow sense key specific data .....	55
4.5.2.5 Field replaceable unit sense data descriptor .....	56
4.5.2.6 Vendor specific sense data descriptors .....	56
4.5.3 Fixed format sense data .....	57
4.5.4 Current errors .....	58
4.5.5 Deferred errors .....	58
4.5.6 Sense key and additional sense code definitions .....	59
4.6 Secure random numbers .....	76
5 Model common to all device types .....	78
5.1 Introduction to the model common to all device types .....	78
5.2 Important commands for all SCSI device servers .....	78
5.2.1 Commands implemented by all SCSI device servers .....	78
5.2.2 Commands recommended for all SCSI device servers .....	78
5.2.3 Using the INQUIRY command .....	78
5.2.4 Using the REPORT LUNS command .....	78
5.2.5 Using the TEST UNIT READY command .....	78
5.2.6 Using the REQUEST SENSE command .....	79
5.3 Implicit head of queue .....	79
5.4 Parameter rounding .....	79
5.5 Parsing variable-length parameter lists and parameter data .....	79
5.6 Self-test operations .....	80
5.6.1 Default self-test .....	80
5.6.2 The short and extended self-tests .....	80
5.6.3 Self-test modes .....	81
5.6.3.1 Self-test modes overview .....	81
5.6.3.2 Foreground mode .....	81
5.6.3.3 Background mode .....	81
5.6.3.4 Features common to foreground and background self-test modes .....	82
5.7 Reservations .....	84
5.7.1 Persistent Reservations overview .....	84
5.7.2 Third party persistent reservations .....	88
5.7.3 Exceptions to SPC-2 RESERVE and RELEASE behavior .....	88
5.7.4 Persistent reservations interactions with iSCSI SA creation .....	89
5.7.5 Preserving persistent reservations and registrations .....	89
5.7.5.1 Preserving persistent reservations and registrations through power loss .....	89
5.7.5.2 Nonvolatile memory considerations for preserving persistent reservations and registrations .....	90

5.7.6 Finding persistent reservations and reservation keys .....	90
5.7.6.1 Summary of commands for finding persistent reservations and reservation keys .....	90
5.7.6.2 Reporting reservation keys .....	90
5.7.6.3 Reporting the persistent reservation .....	91
5.7.6.4 Reporting full status .....	91
5.7.7 Registering .....	91
5.7.8 Registering and moving the reservation .....	95
5.7.9 Reserving .....	96
5.7.10 Persistent reservation holder .....	97
5.7.11 Releasing persistent reservations and removing registrations .....	97
5.7.11.1 Summary of service actions that release persistent reservations and remove registrations .....	97
5.7.11.2 Releasing .....	99
5.7.11.3 Unregistering .....	99
5.7.11.4 Preempting .....	100
5.7.11.4.1 Overview .....	100
5.7.11.4.2 Failed persistent reservation preempt .....	102
5.7.11.4.3 Preempting persistent reservations and registration handling .....	102
5.7.11.4.4 Removing registrations .....	103
5.7.11.5 Preempting and aborting .....	103
5.7.11.6 Clearing .....	104
5.8 Multiple target port and initiator port behavior .....	105
5.9 Target port group access states .....	105
5.9.1 Target port group access overview .....	105
5.9.2 Asymmetric logical unit access .....	105
5.9.2.1 Introduction to asymmetric logical unit access .....	105
5.9.2.2 Explicit and implicit asymmetric logical unit access .....	107
5.9.2.3 Discovery of asymmetric logical unit access behavior .....	107
5.9.2.4 Target port asymmetric access states .....	107
5.9.2.4.1 Target port asymmetric access states overview .....	107
5.9.2.4.2 Active/optimized state .....	107
5.9.2.4.3 Active/non-optimized state .....	108
5.9.2.4.4 Standby state .....	108
5.9.2.4.5 Unavailable state .....	109
5.9.2.4.6 Offline state .....	109
5.9.2.5 Transitions between target port asymmetric access states .....	109
5.9.2.6 Preference Indicator .....	111
5.9.2.7 Implicit asymmetric logical units access management .....	111
5.9.2.8 Explicit asymmetric logical units access management .....	111
5.9.2.9 Behavior after power on, hard reset, logical unit reset, and I_T nexus loss .....	112
5.9.2.10 Behavior of target ports that are not accessible from the service delivery subsystem .....	112
5.9.3 Symmetric logical unit access .....	112
5.10 Power conditions .....	112
5.10.1 Power conditions overview .....	112
5.10.2 Power condition state machine .....	113
5.10.2.1 Power condition state machine overview .....	113
5.10.2.2 PC0:Powered_on state .....	114
5.10.2.3 PC1:Active state .....	114
5.10.2.4 PC2:Idle state .....	115
5.10.2.5 PC3:Standby state .....	115
5.11 Medium auxiliary memory .....	116
5.12 Error history .....	117
5.12.1 Error history overview .....	117
5.12.2 Retrieving error history with the READ BUFFER command .....	117
5.12.3 Adding application client error history with the WRITE BUFFER command .....	119

5.12.4 Clearing error history with the WRITE BUFFER command.....	119
5.13 Device clocks.....	120
5.14 Security Features .....	121
5.14.1 Security goals and threat model.....	121
5.14.1.1 Overview.....	121
5.14.1.2 Security goals .....	121
5.14.1.3 Threat model .....	122
5.14.1.4 Types of attacks .....	122
5.14.1.5 SCSI security considerations.....	123
5.14.2 Security associations.....	124
5.14.2.1 Principles of SAs .....	124
5.14.2.2 SA parameters.....	125
5.14.2.3 Creating an SA .....	128
5.14.3 Key derivation functions .....	128
5.14.3.1 Overview.....	128
5.14.3.2 IKEv2-based iterative KDF .....	129
5.14.3.3 HMAC-based KDFs .....	129
5.14.3.4 AES-XCBC-PRF-128 IKEv2-based iterative KDF .....	130
5.14.4 Using IKEv2-SCSI to create a security association.....	131
5.14.4.1 Overview.....	131
5.14.4.2 IKEv2-SCSI Protocol summary .....	134
5.14.4.3 Handling of the Certificate Request payload and the Certificate payload .....	137
5.14.4.4 Summary of IKEv2-SCSI shared keys nomenclature and shared key sizes .....	138
5.14.4.5 IKEv2-SCSI usage of pre-shared keys.....	139
5.14.4.6 Constraints on skipping the Authentication step.....	140
5.14.4.7 Device Server Capabilities step.....	140
5.14.4.8 IKEv2-SCSI Key Exchange step .....	142
5.14.4.8.1 Overview.....	142
5.14.4.8.2 Key Exchange step SECURITY PROTOCOL OUT command.....	142
5.14.4.8.3 Key Exchange step SECURITY PROTOCOL IN command .....	143
5.14.4.8.4 Key Exchange step completion .....	144
5.14.4.8.5 After the Key Exchange step .....	144
5.14.4.9 IKEv2-SCSI Authentication step.....	145
5.14.4.9.1 Overview.....	145
5.14.4.9.2 Authentication step SECURITY PROTOCOL OUT command .....	145
5.14.4.9.3 Authentication step SECURITY PROTOCOL IN command .....	146
5.14.4.10 Generating shared keys .....	148
5.14.4.10.1 Overview.....	148
5.14.4.10.2 Generating shared keys when the Authentication step is skipped .....	148
5.14.4.10.3 Generating shared keys when the Authentication step is processed .....	148
5.14.4.10.4 Initializing shared key generation .....	149
5.14.4.10.4.1 Initializing for SA creation shared key generation .....	149
5.14.4.10.4.2 Initializing for generation of shared keys used by the created SA.....	149
5.14.4.10.5 Generating shared keys used for SA management.....	150
5.14.4.10.6 Generating shared keys for use by the created SA.....	151
5.14.4.11 IKEv2-SCSI SA generation.....	151
5.14.4.12 Abandoning an IKEv2-SCSI CCS.....	153
5.14.4.13 Deleting an IKEv2-SCSI SA .....	154
5.14.5 Security progress indication .....	154
5.14.6 Command security.....	155
5.14.6.1 Overview.....	155
5.14.6.2 Secure CDB Originator class.....	155
5.14.6.3 Secure CDB Processor class .....	155
5.14.6.4 Enforcement Manager class.....	156

5.14.6.5 Security Manager class .....	156
5.14.6.6 The relationship between SAs and command security .....	157
5.14.6.7 Command security techniques .....	157
5.14.6.8 Capability-based command security technique .....	157
5.14.6.8.1 Overview .....	157
5.14.6.8.2 Security Manager class .....	160
5.14.6.8.3 CbCS Management Device Server class .....	161
5.14.6.8.3.1 CbCS Management Device Server class overview .....	161
5.14.6.8.3.2 Decision Database attribute .....	161
5.14.6.8.4 CbCS Management Application Client class .....	161
5.14.6.8.5 Secure CDB Originator class .....	161
5.14.6.8.6 Secure CDB Processor class .....	162
5.14.6.8.7 Enforcement Manager class .....	162
5.14.6.8.8 CbCS methods .....	163
5.14.6.8.8.1 Overview .....	163
5.14.6.8.8.2 The BASIC CbCS method .....	163
5.14.6.8.8.3 The CAPKEY CbCS method .....	164
5.14.6.8.9 CbCS trust assumptions .....	164
5.14.6.8.10 CbCS security tokens .....	165
5.14.6.8.11 CbCS shared keys .....	166
5.14.6.8.11.1 Overview .....	166
5.14.6.8.11.2 CbCS shared key identifiers .....	167
5.14.6.8.11.3 Specifying which CbCS shared key to change .....	168
5.14.6.8.11.4 Updating a CbCS master key .....	168
5.14.6.8.11.5 Changing a CbCS working keys .....	168
5.14.6.8.12 CbCS credentials .....	169
5.14.6.8.12.1 Overview .....	169
5.14.6.8.12.2 CbCS capability key computations for the secure CDB originator .....	169
5.14.6.8.12.3 CbCS capability key computations for general use .....	169
5.14.6.8.13 CbCS capability descriptors .....	170
5.14.6.8.13.1 Overview .....	170
5.14.6.8.13.2 CbCS extension descriptor validation .....	171
5.14.6.8.13.3 CAPKEY CbCS method capability integrity validation .....	171
5.14.6.8.14 Association between commands and permission bits .....	172
5.14.6.8.15 CbCS parameters .....	175
5.14.6.8.15.1 Overview .....	175
5.14.6.8.16 CbCS extension descriptor format .....	177
5.14.7 ESP-SCSI for parameter data .....	178
5.14.7.1 Overview .....	178
5.14.7.2 ESP-SCSI required inputs .....	178
5.14.7.3 ESP-SCSI data format before encryption and after decryption .....	179
5.14.7.4 ESP-SCSI outbound data descriptors .....	180
5.14.7.4.1 Overview .....	180
5.14.7.4.2 ESP-SCSI CDBs or data-out buffer parameter lists including a descriptor length .....	181
5.14.7.4.3 ESP-SCSI data-out buffer parameter lists for externally specified descriptor length .....	184
5.14.7.5 ESP-SCSI data-in buffer parameter data descriptors .....	185
5.14.7.5.1 Overview .....	185
5.14.7.5.2 ESP-SCSI data-in buffer parameter data including a descriptor length .....	186
5.14.7.5.3 ESP-SCSI data-in buffer parameter data for externally specified descriptor length .....	189
5.14.8 Security algorithm codes .....	191
5.15 Identifying information .....	193
5.16 Downloading and activating microcode .....	193

6 Commands for all device types .....	198
6.1 Summary of commands for all device types .....	198
6.2 CHANGE ALIASES command .....	200
6.2.1 CHANGE ALIASES command introduction .....	200
6.2.2 Alias entry format .....	202
6.2.3 Alias designation validation .....	203
6.2.4 Alias entry protocol independent designations .....	203
6.2.4.1 Alias entry protocol independent designations overview .....	203
6.2.4.2 NULL DESIGNATION alias format .....	203
6.3 EXTENDED COPY command .....	204
6.3.1 EXTENDED COPY command introduction .....	204
6.3.2 Errors detected before starting processing of the segment descriptors .....	207
6.3.3 Errors detected during processing of segment descriptors .....	207
6.3.4 Abort task management functions .....	209
6.3.5 Descriptor type codes .....	210
6.3.6 Target descriptors .....	212
6.3.6.1 Target descriptors introduction .....	212
6.3.6.2 Identification descriptor target descriptor format .....	214
6.3.6.3 Alias target descriptor format .....	215
6.3.6.4 Device type specific target descriptor parameters for block device types .....	216
6.3.6.5 Device type specific target descriptor parameters for sequential-access device types .....	216
6.3.6.6 Device type specific target descriptor parameters for processor device types .....	217
6.3.7 Segment descriptors .....	218
6.3.7.1 Segment descriptors introduction .....	218
6.3.7.2 Segment descriptor processing .....	219
6.3.7.3 Block device to stream device operations .....	222
6.3.7.4 Stream device to block device operations .....	223
6.3.7.5 Block device to block device operations .....	224
6.3.7.6 Stream device to stream device operations .....	226
6.3.7.7 Inline data to stream device operation .....	228
6.3.7.8 Embedded data to stream device operation .....	229
6.3.7.9 Stream device to discard operation .....	230
6.3.7.10 Verify device operation .....	231
6.3.7.11 Block device with offset to stream device operation .....	232
6.3.7.12 Stream device to block device with offset operation .....	233
6.3.7.13 Block device with offset to block device with offset operation .....	234
6.3.7.14 Write filemarks operation .....	235
6.3.7.15 Space operation .....	236
6.3.7.16 Locate operation .....	237
6.3.7.17 Tape device image copy operation .....	238
6.3.7.18 Register persistent reservation key operation .....	239
6.3.7.19 Third party persistent reservations source I_T nexus .....	240
6.4 INQUIRY command .....	242
6.4.1 INQUIRY command introduction .....	242
6.4.2 Standard INQUIRY data .....	244
6.4.3 SCSI Parallel Interface specific INQUIRY data .....	257
6.4.4 Vital product data .....	258
6.5 LOG SELECT command .....	259
6.5.1 Introduction .....	259
6.5.2 Processing LOG SELECT when the parameter list length is zero .....	261
6.6 LOG SENSE command .....	264
6.7 MANAGEMENT PROTOCOL IN command .....	266
6.7.1 MANAGEMENT PROTOCOL IN command description .....	266
6.7.2 Management protocol information description .....	267



6.7.2.1 Overview.....	267
6.7.2.2 CDB description.....	267
6.7.2.3 Supported management protocols list description.....	268
6.8 MANAGEMENT PROTOCOL OUT command .....	269
6.9 MODE SELECT(6) command.....	270
6.10 MODE SELECT(10) command.....	272
6.11 MODE SENSE(6) command .....	272
6.11.1 MODE SENSE(6) command introduction .....	272
6.11.2 Current values .....	274
6.11.3 Changeable values.....	274
6.11.4 Default values.....	274
6.11.5 Saved values .....	274
6.11.6 Initial responses.....	275
6.12 MODE SENSE(10) command .....	275
6.13 PERSISTENT RESERVE IN command .....	276
6.13.1 PERSISTENT RESERVE IN command introduction .....	276
6.13.2 READ KEYS service action .....	277
6.13.3 READ RESERVATION service action .....	277
6.13.3.1 READ RESERVATION service action introduction .....	277
6.13.3.2 Format of PERSISTENT RESERVE IN parameter data for READ RESERVATION .....	278
6.13.3.3 Persistent reservations scope .....	279
6.13.3.4 Persistent reservations type .....	279
6.13.4 REPORT CAPABILITIES service action .....	280
6.13.5 READ FULL STATUS service action.....	283
6.14 PERSISTENT RESERVE OUT command .....	285
6.14.1 PERSISTENT RESERVE OUT command introduction .....	285
6.14.2 PERSISTENT RESERVE OUT service actions .....	287
6.14.3 Basic PERSISTENT RESERVE OUT parameter list.....	288
6.14.4 PERSISTENT RESERVE OUT command with REGISTER AND MOVE service action parameters .....	291
6.15 READ ATTRIBUTE command.....	293
6.15.1 READ ATTRIBUTE command introduction .....	293
6.15.2 ATTRIBUTE VALUES service action .....	294
6.15.3 ATTRIBUTE LIST service action .....	295
6.15.4 VOLUME LIST service action.....	296
6.15.5 PARTITION LIST service action.....	296
6.16 READ BUFFER command .....	297
6.16.1 READ BUFFER command introduction.....	297
6.16.2 Combined header and data mode (00h).....	298
6.16.3 Vendor specific mode (01h).....	298
6.16.4 Data mode (02h).....	298
6.16.5 Descriptor mode (03h).....	299
6.16.6 Read data from echo buffer mode (0Ah) .....	299
6.16.7 Echo buffer descriptor mode (0Bh).....	300
6.16.8 Enable expander communications protocol and Echo buffer (1Ah) .....	300
6.16.9 Error history mode (1Ch) .....	301
6.16.9.1 Error history overview.....	301
6.16.9.2 Error history directory .....	302
6.16.9.3 Error history data buffer.....	304
6.16.9.4 Clear error history I_T nexus .....	304
6.16.9.5 Clear error history I_T nexus and release snapshot.....	304
6.17 READ MEDIA SERIAL NUMBER command .....	305
6.18 RECEIVE COPY RESULTS command .....	306
6.18.1 RECEIVE COPY RESULTS command introduction.....	306
6.18.2 COPY STATUS service action .....	307

6.18.3 RECEIVE DATA service action .....	309
6.18.4 OPERATING PARAMETERS service action.....	310
6.18.5 FAILED SEGMENT DETAILS service action .....	313
6.19 RECEIVE CREDENTIAL command .....	314
6.19.1 RECEIVE CREDENTIAL command description.....	314
6.19.1.1 Overview.....	314
6.19.1.2 CbCS logical unit credential request descriptor.....	316
6.19.1.3 CbCS logical unit and volume credential request descriptor .....	316
6.19.2 RECEIVE CREDENTIAL parameter data.....	317
6.19.2.1 RECEIVE CREDENTIAL parameter data encryption .....	317
6.19.2.2 RECEIVE CREDENTIAL decrypted parameter data .....	317
6.19.2.3 CbCS capability descriptor .....	318
6.19.2.3.1 Overview.....	318
6.19.2.3.2 Logical unit designation descriptor format.....	321
6.19.2.3.3 Volume designation descriptors .....	321
6.20 RECEIVE DIAGNOSTIC RESULTS command .....	321
6.21 REPORT ALIASES command.....	322
6.22 REPORT IDENTIFYING INFORMATION command.....	324
6.22.1 REPORT IDENTIFYING INFORMATION command overview.....	324
6.22.2 IDENTIFYING INFORMATION parameter data .....	325
6.22.3 IDENTIFYING INFORMATION SUPPORTED parameter data .....	326
6.23 REPORT LUNS command .....	327
6.24 REPORT PRIORITY command.....	329
6.25 REPORT SUPPORTED OPERATION CODES command .....	331
6.25.1 REPORT SUPPORTED OPERATION CODES command introduction.....	331
6.25.2 All_commands parameter data format .....	333
6.25.3 One_command parameter data format .....	334
6.25.4 Command timeouts descriptor.....	335
6.25.4.1 Overview.....	335
6.25.4.2 WRITE BUFFER command timeouts descriptor COMMAND SPECIFIC field usage.....	337
6.26 REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS command .....	337
6.27 REPORT TARGET PORT GROUPS command.....	339
6.28 REPORT TIMESTAMP command.....	343
6.29 REQUEST SENSE command .....	344
6.30 SECURITY PROTOCOL IN command.....	347
6.31 SECURITY PROTOCOL OUT command.....	349
6.32 SEND DIAGNOSTIC command .....	350
6.33 SET IDENTIFYING INFORMATION command.....	353
6.34 SET PRIORITY command.....	355
6.35 SET TARGET PORT GROUPS command.....	358
6.36 SET TIMESTAMP command.....	361
6.37 TEST UNIT READY command.....	362
6.38 WRITE ATTRIBUTE command .....	363
6.39 WRITE BUFFER command.....	365
6.39.1 WRITE BUFFER command introduction .....	365
6.39.2 Combined header and data mode (00h).....	366
6.39.3 Vendor specific mode (01h).....	366
6.39.4 Data mode (02h).....	367
6.39.5 Download microcode and activate mode (04h) .....	367
6.39.6 Download microcode, save, and activate mode (05h).....	367
6.39.7 Download microcode with offsets and activate mode (06h) .....	367
6.39.8 Download microcode with offsets, save, and activate mode (07h).....	368
6.39.9 Write data to echo buffer mode (0Ah) .....	368
6.39.10 Download microcode with offsets, save, and defer activate mode (0Eh) .....	368

6.39.11 Activate deferred microcode mode (0Fh) .....	369
6.39.12 Enable expander communications protocol and Echo buffer mode (1Ah) .....	369
6.39.13 Disable expander communications protocol mode (1Bh) .....	369
6.39.14 Download application client error history mode (1Ch) .....	369
7 Parameters for all device types .....	373
7.1 Diagnostic parameters.....	373
7.1.1 Diagnostic page format and page codes for all device types .....	373
7.1.2 Protocol Specific.....	374
7.1.3 Supported Diagnostic Pages .....	375
7.2 Log parameters .....	376
7.2.1 Log page structure and log parameter structure for all device types.....	376
7.2.1.1 Log page structure.....	376
7.2.1.2 Log parameter structure .....	378
7.2.1.2.1 Introduction.....	378
7.2.1.2.2 Parameter control byte .....	379
7.2.1.2.2.1 Introduction.....	379
7.2.1.2.2.2 Parameter control byte values for data counter parameters .....	382
7.2.1.2.2.3 Parameter control byte values for list parameters .....	383
7.2.2 Log page codes for all device types .....	384
7.2.3 Application Client log page .....	385
7.2.4 Buffer Over-Run/Under-Run log page .....	386
7.2.5 Error Counter log pages .....	387
7.2.6 Informational Exceptions log page .....	388
7.2.7 Last n Deferred Errors or Asynchronous Events log page .....	389
7.2.8 Last n Error Events log page .....	389
7.2.9 Non-Medium Error log page .....	390
7.2.10 Protocol Specific Port log page .....	390
7.2.11 Self-Test Results log page .....	392
7.2.12 Start-Stop Cycle Counter log page.....	395
7.2.13 Statistics and Performance log pages .....	397
7.2.13.1 Statistics and Performance log pages overview.....	397
7.2.13.2 General Statistics and Performance log page .....	398
7.2.13.3 Group Statistics and Performance (n) log page .....	404
7.2.14 Supported Log Pages log page .....	408
7.2.15 Supported Log Pages and Subpages log page.....	409
7.2.16 Supported Subpages log page .....	410
7.2.17 Temperature log page .....	411
7.3 Medium auxiliary memory attributes.....	413
7.3.1 Attribute format.....	413
7.3.2 Attribute identifier values .....	414
7.3.2.1 Attribute identifier values overview .....	414
7.3.2.2 Device type attributes .....	415
7.3.2.3 Medium type attributes .....	421
7.3.2.4 Host type attributes.....	422
7.4 Mode parameters .....	424
7.4.1 Mode parameters overview .....	424
7.4.2 Mode parameter list format.....	424
7.4.3 Mode parameter header formats .....	424
7.4.4 Mode parameter block descriptor formats .....	426
7.4.4.1 General block descriptor format .....	426
7.4.5 Mode page and subpage formats and page codes .....	427
7.4.6 Control mode page.....	429
7.4.7 Control Extension mode page .....	434

7.4.8 Disconnect-Reconnect mode page .....	435
7.4.9 Extended mode page .....	438
7.4.10 Extended Device-Type Specific mode page.....	438
7.4.11 Informational Exceptions Control mode page.....	439
7.4.12 Power Condition mode page .....	442
7.4.13 Protocol Specific Logical Unit mode page .....	443
7.4.14 Protocol Specific Port mode page .....	444
7.5 Protocol specific parameters .....	446
7.5.1 Protocol specific parameters introduction.....	446
7.5.2 Alias entry protocol specific designations.....	446
7.5.2.1 Introduction to alias entry protocol specific designations .....	446
7.5.2.2 Fibre Channel specific alias entry designations .....	447
7.5.2.2.1 Introduction to Fibre Channel specific alias entry designations.....	447
7.5.2.2.2 Fibre Channel world wide port name alias entry designation .....	447
7.5.2.2.3 Fibre Channel world wide port name with N_Port checking alias entry designation .....	448
7.5.2.3 RDMA specific alias entry designations .....	448
7.5.2.3.1 Introduction to RDMA specific alias entry designations.....	448
7.5.2.3.2 RDMA target port identifier alias entry designation .....	449
7.5.2.3.3 InfiniBand global identifier with target port identifier checking alias entry designation .....	449
7.5.2.4 Internet SCSI specific alias entry designations .....	450
7.5.2.4.1 Introduction to Internet SCSI specific alias entry designations.....	450
7.5.2.4.2 iSCSI name alias entry designation.....	450
7.5.2.4.3 iSCSI name with binary IPv4 address alias entry designation .....	451
7.5.2.4.4 iSCSI name with IPname alias entry designation.....	452
7.5.2.4.5 iSCSI name with binary IPv6 address alias entry designation .....	453
7.5.3 EXTENDED COPY protocol specific target descriptors .....	454
7.5.3.1 Introduction to EXTENDED COPY protocol specific target descriptors .....	454
7.5.3.2 Fibre Channel N_Port_Name EXTENDED COPY target descriptor format .....	454
7.5.3.3 Fibre Channel N_Port_ID EXTENDED COPY target descriptor format .....	455
7.5.3.4 Fibre Channel N_Port_ID with N_Port_Name checking EXTENDED COPY target descriptor format.....	456
7.5.3.5 SCSI Parallel T_L EXTENDED COPY target descriptor format .....	457
7.5.3.6 IEEE 1394 EUI-64 EXTENDED COPY target descriptor format .....	458
7.5.3.7 RDMA EXTENDED COPY target descriptor format .....	459
7.5.3.8 iSCSI binary IPv4 address EXTENDED COPY target descriptor format.....	460
7.5.3.9 SAS serial SCSI protocol target descriptor format .....	461
7.5.4 TransportID identifiers .....	461
7.5.4.1 Overview of TransportID identifiers .....	461
7.5.4.2 TransportID for initiator ports using SCSI over Fibre Channel .....	462
7.5.4.3 TransportID for initiator ports using a parallel SCSI bus .....	463
7.5.4.4 TransportID for initiator ports using SCSI over IEEE 1394.....	463
7.5.4.5 TransportID for initiator ports using SCSI over an RDMA interface .....	464
7.5.4.6 TransportID for initiator ports using SCSI over iSCSI.....	464
7.5.4.7 TransportID for initiator ports using SCSI over SAS Serial SCSI Protocol.....	466
7.6 Security protocol parameters.....	467
7.6.1 Security protocol information description.....	467
7.6.1.1 Overview.....	467
7.6.1.2 CDB description.....	467
7.6.1.3 Supported security protocols list description .....	468
7.6.1.4 Certificate data description .....	469
7.6.1.4.1 Certificate overview .....	469
7.6.1.4.2 Public Key certificate description.....	469
7.6.1.4.3 Attribute certificate description .....	469
7.6.2 SA creation capabilities .....	470
7.6.2.1 Overview.....	470

7.6.2.2 SA creation capabilities CDB description .....	470
7.6.2.3 SA creation capabilities parameter data formats .....	471
7.6.2.3.1 Supported device server capabilities formats parameter data format .....	471
7.6.2.3.2 IKEv2-SCSI device server capabilities parameter data format .....	471
7.6.3 IKEv2-SCSI .....	472
7.6.3.1 Overview .....	472
7.6.3.2 IKEv2-SCSI SECURITY PROTOCOL IN CDB description .....	472
7.6.3.3 IKEv2-SCSI SECURITY PROTOCOL OUT CDB description .....	473
7.6.3.4 IKEv2-SCSI parameter data format .....	474
7.6.3.5 IKEv2-SCSI payloads .....	482
7.6.3.5.1 IKEv2-SCSI payload format .....	482
7.6.3.5.2 No Next payload .....	483
7.6.3.5.3 Key Exchange payload .....	483
7.6.3.5.4 Identification – Application Client payload and Identification – Device Server payload .....	484
7.6.3.5.5 Certificate payload and Certificate Request payload .....	485
7.6.3.5.6 Authentication payload .....	486
7.6.3.5.7 Nonce payload .....	489
7.6.3.5.8 Notify payload .....	490
7.6.3.5.9 Delete payload .....	491
7.6.3.5.10 Encrypted payload .....	492
7.6.3.5.10.1 Combined mode encryption .....	492
7.6.3.5.10.2 Encrypted payload introduction .....	493
7.6.3.5.10.3 IKEv2-SCSI AAD .....	495
7.6.3.5.10.4 Processing a received Encrypted payload .....	496
7.6.3.5.11 IKEv2-SCSI SA Creation Capabilities payload .....	498
7.6.3.5.12 IKEv2-SCSI SA Cryptographic Algorithms payload .....	499
7.6.3.5.13 IKEv2-SCSI SAUT Cryptographic Algorithms payload .....	501
7.6.3.5.14 IKEv2-SCSI Timeout Values payload .....	502
7.6.3.6 IKEv2-SCSI cryptographic algorithm descriptors .....	503
7.6.3.6.1 Overview .....	503
7.6.3.6.2 Encryption algorithm (ENCR) IKEv2-SCSI cryptographic algorithm descriptor .....	505
7.6.3.6.3 Pseudo-random function (PRF) IKEv2-SCSI cryptographic algorithm descriptor .....	507
7.6.3.6.4 Integrity algorithm (INTEG) IKEv2-SCSI cryptographic algorithm descriptor .....	509
7.6.3.6.5 Diffie-Hellman group (D-H) IKEv2-SCSI cryptographic algorithm descriptor .....	510
7.6.3.6.6 IKEv2-SCSI authentication algorithm IKEv2-SCSI cryptographic algorithm descriptor .....	512
7.6.3.7 Errors in IKEv2-SCSI security protocol commands .....	515
7.6.3.8 Errors in IKEv2-SCSI security protocol parameter data .....	517
7.6.3.8.1 Overview .....	517
7.6.3.8.2 Errors with high denial of service attack potential .....	517
7.6.3.8.3 Errors with minimal denial of service attack potential .....	518
7.6.3.9 Translating IKEv2 errors .....	519
7.6.4 CbCS security protocol .....	520
7.6.4.1 Overview .....	520
7.6.4.2 CbCS SECURITY PROTOCOL IN CDB description .....	520
7.6.4.3 CbCS SECURITY PROTOCOL IN parameter data .....	521
7.6.4.3.1 Supported CbCS SECURITY PROTOCOL IN Pages CbCS page .....	521
7.6.4.3.2 Supported CbCS SECURITY PROTOCOL OUT pages CbCS page .....	522
7.6.4.3.3 Unchangeable CbCS Parameters CbCS page .....	523
7.6.4.3.4 Security Token CbCS page .....	525
7.6.4.3.5 Current CbCS Parameters CbCS page .....	525
7.6.4.3.6 Set Master Key – Seed Exchange CbCS page .....	527
7.6.4.4 CbCS SECURITY PROTOCOL OUT CDB description .....	529
7.6.4.5 CbCS SECURITY PROTOCOL OUT parameter data .....	530
7.6.4.5.1 Set Policy Access Tag CbCS page .....	530

7.6.4.5.2 Set Minimum CbCS Method CbCS page .....	530
7.6.4.5.3 Invalidate Key CbCS page .....	531
7.6.4.5.4 Set Key CbCS page .....	533
7.6.4.5.5 Set Master Key – Seed Exchange CbCS page .....	534
7.6.4.5.6 Set Master Key – Change Master Key CbCS page.....	535
7.7 Vital product data parameters .....	537
7.7.1 Vital product data parameters overview and page codes.....	537
7.7.2 ASCII Information VPD page.....	538
7.7.3 Device Identification VPD page .....	539
7.7.3.1 Device Identification VPD page overview.....	539
7.7.3.2 Device designation descriptor requirements .....	541
7.7.3.2.1 Designation descriptors for logical units other than well known logical units .....	541
7.7.3.2.2 Designation descriptors for well known logical units .....	542
7.7.3.2.3 Designation descriptors for SCSI target ports .....	542
7.7.3.2.3.1 Relative target port identifiers.....	542
7.7.3.2.3.2 Target port names or identifiers.....	542
7.7.3.2.4 Designation descriptors for SCSI target devices .....	542
7.7.3.3 Vendor specific designator format.....	543
7.7.3.4 T10 vendor ID based designator format .....	543
7.7.3.5 EUI-64 based designator format.....	544
7.7.3.5.1 EUI-64 based designator format overview .....	544
7.7.3.5.2 EUI-64 designator format .....	544
7.7.3.5.3 EUI-64 based 12-byte designator format.....	545
7.7.3.5.4 EUI-64 based 16-byte designator format.....	545
7.7.3.6 NAA designator format .....	546
7.7.3.6.1 NAA identifier basic format .....	546
7.7.3.6.2 NAA IEEE Extended designator format.....	546
7.7.3.6.3 NAA Locally Assigned designator format .....	547
7.7.3.6.4 NAA IEEE Registered designator format.....	547
7.7.3.6.5 NAA IEEE Registered Extended designator format .....	548
7.7.3.7 Relative target port designator format .....	548
7.7.3.8 Target port group designator format.....	549
7.7.3.9 Logical unit group designator format .....	549
7.7.3.10 MD5 logical unit designator format .....	550
7.7.3.11 SCSI name string designator format .....	551
7.7.4 Extended INQUIRY Data VPD page .....	552
7.7.5 Management Network Addresses VPD page .....	554
7.7.6 Mode Page Policy VPD page .....	556
7.7.7 SCSI Ports VPD page .....	558
7.7.8 Protocol Specific Logical Unit Information VPD page.....	561
7.7.9 Protocol Specific Port Information VPD page.....	562
7.7.10 Software Interface Identification VPD page.....	564
7.7.11 Supported VPD Pages VPD page .....	565
7.7.12 Unit Serial Number VPD page.....	565
8 Well known logical units .....	566
8.1 Model for well known logical units .....	566
8.2 REPORT LUNS well known logical unit .....	566
8.3 ACCESS CONTROLS well known logical unit .....	567
8.3.1 Access controls model.....	567
8.3.1.1 Access controls commands.....	567
8.3.1.2 Access controls overview .....	567
8.3.1.3 The access control list (ACL).....	568
8.3.1.3.1 ACL overview .....	568

8.3.1.3.2 Access identifiers.....	569
8.3.1.3.3 Logical unit access control descriptors.....	570
8.3.1.4 Managing the ACL.....	570
8.3.1.4.1 ACL management overview .....	570
8.3.1.4.2 Authorizing ACL management.....	570
8.3.1.4.3 Identifying logical units during ACL management .....	571
8.3.1.4.4 Tracking changes in logical unit identification .....	571
8.3.1.5 Enrolling AccessIDs.....	572
8.3.1.5.1 Enrollment states.....	572
8.3.1.5.1.1 Summary of enrollment states.....	572
8.3.1.5.1.2 Not-enrolled state .....	572
8.3.1.5.1.3 Enrolled state.....	573
8.3.1.5.1.4 Pending-enrolled state.....	574
8.3.1.5.2 ACL LUN conflict resolution.....	574
8.3.1.6 Granting and revoking access rights .....	574
8.3.1.6.1 Non-proxy access rights .....	574
8.3.1.6.2 Proxy access .....	575
8.3.1.6.2.1 Proxy tokens.....	575
8.3.1.6.2.2 Proxy LUNs .....	575
8.3.1.7 Verifying access rights.....	576
8.3.1.8 The management identifier key .....	577
8.3.1.8.1 Management identifier key usage.....	577
8.3.1.8.2 Overriding the management identifier key.....	578
8.3.1.8.2.1 The OVERRIDE MGMT ID KEY service action.....	578
8.3.1.8.2.2 The override lockout timer .....	578
8.3.1.9 Reporting access control information .....	579
8.3.1.10 Access controls log.....	579
8.3.1.11 Interactions of access controls and other features .....	580
8.3.1.11.1 Task set management and access controls .....	580
8.3.1.11.2 Existing reservations and ACL changes.....	580
8.3.1.12 Access controls information persistence and memory usage requirements .....	581
8.3.1.13 Access identifier formats .....	582
8.3.1.13.1 Access identifier type.....	582
8.3.1.13.2 AccessID access identifiers.....	582
8.3.2 ACCESS CONTROL IN command.....	583
8.3.2.1 ACCESS CONTROL IN introduction .....	583
8.3.2.2 REPORT ACL service action.....	584
8.3.2.2.1 REPORT ACL introduction .....	584
8.3.2.2.2 REPORT ACL parameter data format .....	585
8.3.2.2.2.1 REPORT ACL parameter data introduction.....	585
8.3.2.2.2.2 Granted ACL data page format .....	586
8.3.2.2.2.3 Granted All ACL data page format .....	588
8.3.2.2.2.4 Proxy Tokens ACL data page format .....	589
8.3.2.3 REPORT LU DESCRIPTORS service action .....	590
8.3.2.3.1 REPORT LU DESCRIPTORS introduction .....	590
8.3.2.3.2 REPORT LU DESCRIPTORS parameter data format .....	591
8.3.2.4 REPORT ACCESS CONTROLS LOG service action .....	595
8.3.2.4.1 REPORT ACCESS CONTROLS LOG introduction.....	595
8.3.2.4.2 REPORT ACCESS CONTROLS LOG parameter data format.....	596
8.3.2.4.2.1 REPORT ACCESS CONTROLS LOG parameter data introduction .....	596
8.3.2.4.2.2 Key Overrides access controls log portion page format.....	597
8.3.2.4.2.3 Invalid Keys access controls log portion page format .....	598
8.3.2.4.2.4 ACL LUN Conflicts access controls log portion page format.....	599
8.3.2.5 REPORT OVERRIDE LOCKOUT TIMER service action .....	600

8.3.2.6 REQUEST PROXY TOKEN service action .....	601
8.3.3 ACCESS CONTROL OUT command.....	603
8.3.3.1 ACCESS CONTROL OUT introduction .....	603
8.3.3.2 MANAGE ACL service action .....	604
8.3.3.2.1 MANAGE ACL introduction .....	604
8.3.3.2.2 The Grant/Revoke ACE page.....	607
8.3.3.2.3 The Grant All ACE page .....	609
8.3.3.2.4 The Revoke Proxy Token ACE page.....	610
8.3.3.2.5 The Revoke All Proxy Tokens ACE page.....	611
8.3.3.3 DISABLE ACCESS CONTROLS service action.....	611
8.3.3.4 ACCESS ID ENROLL service action.....	612
8.3.3.5 CANCEL ENROLLMENT service action .....	613
8.3.3.6 CLEAR ACCESS CONTROLS LOG service action .....	614
8.3.3.7 MANAGE OVERRIDE LOCKOUT TIMER service action.....	615
8.3.3.8 OVERRIDE MGMT ID KEY service action .....	616
8.3.3.9 REVOKE PROXY TOKEN service action.....	617
8.3.3.10 REVOKE ALL PROXY TOKENS service action.....	617
8.3.3.11 ASSIGN PROXY LUN service action .....	618
8.3.3.12 RELEASE PROXY LUN service action .....	619
8.4 TARGET LOG PAGES well known logical unit .....	621
8.5 SECURITY PROTOCOL well known logical unit.....	621
8.6 MANAGEMENT PROTOCOL well known logical unit.....	622
9 Security manager command set .....	623
Annex A (informative) Terminology mapping.....	624
Annex B (Informative) PERSISTENT RESERVE IN/OUT functionality for RESERVE/RELEASE replacement...	625
B.1 Introduction .....	625
B.2 Replacing the reserve/release method with the PERSISTENT RESERVE OUT COMMAND.....	625
B.3 Third party reservations .....	626
Annex C (Informative) Variations between this standard and equivalent security protocols.....	627
C.1 IKEv2 protocol details and variations for IKEv2-SCSI .....	627
C.2 ESP protocol details and variations for ESP-SCSI .....	629
Annex D (informative) Numeric order codes .....	630
D.1 Numeric order codes introduction .....	630
D.2 Additional sense codes .....	630
D.3 Operation codes.....	647
D.3.1 Operation codes.....	647
D.3.2 Additional operation codes for devices with the EncServ bit set to one.....	653
D.3.3 MAINTENANCE (IN) and MAINTENANCE (OUT) service actions .....	654
D.3.4 SERVICE ACTION IN and SERVICE ACTION OUT service actions .....	655
D.3.5 Variable length CDB service action codes .....	656
D.4 Diagnostic page codes.....	657
D.5 Log page codes .....	658
D.6 Mode page codes .....	661
D.7 VPD page codes .....	664
D.8 Version descriptor values.....	666
D.9 T10 IEEE binary identifiers .....	681
Annex E (informative) T10 vendor identification .....	682



## Tables

	Page
1 Numbering conventions examples .....	26
2 Binary power multiplier nomenclature .....	29
3 Class diagram constraints and notes notation .....	30
4 Class diagram multiplicity notation .....	30
5 Class diagram notation for classes .....	31
6 Class diagram notation for associations .....	32
7 Class diagram notation for aggregations .....	33
8 Class diagram notation for generalizations .....	34
9 Class diagram notation for dependencies .....	35
10 Notation for objects .....	35
11 Typical CDB for 6-byte commands .....	37
13 Typical CDB for 12-byte commands .....	38
12 Typical CDB for 10-byte commands .....	38
14 Typical CDB for 16-byte commands .....	39
15 Typical CDB for long LBA 16-byte commands .....	40
16 Typical variable length CDB .....	41
17 Typical variable length CDB for long LBA 32-byte commands .....	42
18 XCDB format .....	43
19 XCDB descriptor format .....	43
20 EXTENSION TYPE field .....	44
21 OPERATION CODE byte .....	44
22 Group Code values .....	45
23 Code set enumeration .....	47
24 Sense data response codes .....	48
25 Descriptor format sense data .....	49
26 Sense data descriptor format .....	50
27 DESCRIPTOR TYPE field .....	50
28 Information sense data descriptor format .....	51
29 Command-specific information sense data descriptor format .....	51
30 Sense key specific sense data descriptor format .....	52
31 Sense key specific sense data descriptor definitions .....	53
32 Field pointer sense key specific data .....	53
33 Actual retry count sense key specific data .....	54
34 Progress indication sense key specific data .....	54
35 Segment pointer sense key specific data .....	55
36 Unit attention condition queue overflow sense key specific data .....	55
37 Field replaceable unit sense data descriptor format .....	56
38 Vendor specific sense data descriptor format .....	56
39 Fixed format sense data .....	57
40 Sense key descriptions .....	59
41 ASC and ASCQ assignments .....	61
42 Exception commands for background self-tests .....	82
43 Self-test mode summary .....	83
44 SPC commands that are allowed in the presence of various reservations .....	85
45 PERSISTENT RESERVE OUT service actions that are allowed in the presence of various reservations .....	88
46 Register behaviors for a REGISTER service action .....	92
47 Register behaviors for a REGISTER AND IGNORE EXISTING KEY service action .....	93
48 I_T Nexuses being registered .....	94
49 Register behaviors for a REGISTER AND MOVE service action .....	95
50 Processing for a released or preempted persistent reservation .....	98
51 Preempting actions .....	100
52 Power Conditions .....	113

53 Types of MAM attributes .....	116
54 MAM attribute states .....	116
55 TIMESTAMP ORIGIN field .....	120
56 TIMESTAMP field format .....	120
57 Minimum SA parameters .....	125
58 USAGE_TYPE SA parameter values .....	127
59 Security protocols that create SAs .....	128
60 Key derivation functions summary .....	128
61 HMAC-based key derivation functions .....	130
62 Hash functions used by HMAC based on KDF_ID .....	130
63 RFC 3566 parameter translations KDF based on AES-XCBC-PRF-128 .....	131
64 IKEv2-SCSI shared key names and SA shared key names .....	138
65 Shared key size determination .....	139
66 Device Server Capabilities step parameter data requirements .....	141
67 IKEv2-SCSI command terminations that do not abandon the CCS, if any .....	153
68 Security Manager class relationships .....	156
69 CbCS methods .....	163
70 CbCS communications trust requirement .....	165
71 Summary of CbCS shared keys .....	167
72 CbCS shared key identifier values .....	167
73 Associations between commands and permissions .....	172
74 Associations between security protocol commands and permissions .....	174
75 Summary of changeable CbCS parameters .....	175
76 CbCS extension descriptor format .....	177
77 ESP-SCSI data format before encryption and after decryption .....	179
78 ESP-SCSI outbound data descriptors .....	180
79 ESP-SCSI CDBs or data-out buffer parameter list descriptor without initialization vector .....	181
80 ESP-SCSI CDBs or data-out buffer full parameter list descriptor .....	182
81 ESP-SCSI data-out buffer parameter list descriptor without length and initialization vector .....	184
82 ESP-SCSI data-out buffer parameter list descriptor without length .....	185
83 ESP-SCSI data-in buffer parameter data descriptors .....	186
84 ESP-SCSI data-in buffer parameter data descriptor without initialization vector .....	186
85 ESP-SCSI data-in buffer full parameter data descriptor .....	187
86 ESP-SCSI data-in buffer parameter data descriptor without length and initialization vector .....	189
87 ESP-SCSI data-in buffer parameter data descriptor without length .....	190
88 Security algorithm codes .....	191
89 Identifying information types .....	193
90 WRITE BUFFER download microcode modes .....	194
91 WRITE BUFFER download microcode field processing .....	195
92 Multiple I_T nexus handling for WRITE BUFFER download microcode modes .....	196
93 Commands for all device types .....	198
94 CHANGE ALIASES command .....	200
95 CHANGE ALIASES parameter list .....	201
96 Alias entry format .....	202
97 Alias entry PROTOCOL IDENTIFIER field .....	202
98 Protocol independent alias entry format codes .....	203
99 EXTENDED COPY command .....	204
100 EXTENDED COPY parameter list .....	205
101 EXTENDED COPY descriptor type codes .....	210
102 Target descriptor format .....	212
103 LU ID TYPE field .....	212
104 Device type specific parameters in target descriptors .....	213
105 Identification descriptor target descriptor format .....	214
106 Alias target descriptor format .....	215

107 Device type specific target descriptor parameters for block device types.....	216
108 Device type specific target descriptor parameters for sequential-access device types .....	216
109 Stream device transfer lengths .....	217
110 Device type specific target descriptor parameters for processor device types .....	217
111 Segment descriptor header.....	218
112 Descriptor Type Code Dependent Copy Manager Processing .....	219
113 PAD and CAT bit definitions .....	221
114 Block device to or from stream device segment descriptor.....	222
115 Block device to block device segment descriptor .....	224
116 Stream device to stream device segment descriptor .....	226
117 Inline data to stream device segment descriptor .....	228
118 Embedded data to stream device segment descriptor.....	229
119 Stream device to discard segment descriptor .....	230
120 Verify device operation segment descriptor .....	231
121 Block device with offset to or from stream device segment descriptor .....	232
122 Block device with offset to block device with offset segment descriptor .....	234
123 Write filemarks operation segment descriptor.....	235
124 Space operation segment descriptor .....	236
125 Locate operation segment descriptor.....	237
126 Tape device image copy segment descriptor .....	238
127 Register persistent reservation key segment descriptor .....	239
128 Third party persistent reservations source I_T nexus segment descriptor.....	240
129 INQUIRY command .....	242
130 Standard INQUIRY data format .....	244
131 Peripheral qualifier .....	245
132 Peripheral device type .....	245
133 Version .....	246
134 TPGS field.....	247
135 Version descriptor values.....	248
136 SPI-specific standard INQUIRY bits .....	257
137 Maximum logical device configuration table .....	257
138 CLOCKING field .....	258
139 LOG SELECT command.....	259
140 Page control (PC) field.....	260
141 PAGE CODE field and SUBPAGE CODE field .....	261
142 PCR bit, SP bit, and PC field meanings when parameter list length is zero.....	261
143 LOG SENSE command .....	264
144 MANAGEMENT PROTOCOL IN command.....	266
145 MANAGEMENT PROTOCOL field in MANAGEMENT PROTOCOL IN command.....	266
146 MANAGEMENT PROTOCOL SPECIFIC2 field for MANAGEMENT PROTOCOL IN protocol 00h .....	267
147 Supported management protocols MANAGEMENT PROTOCOL IN parameter data .....	268
148 MANAGEMENT PROTOCOL OUT command.....	269
149 MANAGEMENT PROTOCOL field in MANAGEMENT PROTOCOL OUT command.....	269
150 MODE SELECT(6) command .....	270
151 Mode page policies .....	270
152 MODE SELECT(10) command .....	272
153 MODE SENSE(6) command .....	272
154 Page control (PC) field.....	273
155 Mode page code usage for all devices .....	273
156 MODE SENSE(10) command .....	275
157 PERSISTENT RESERVE IN command.....	276
158 PERSISTENT RESERVE IN service action codes .....	276
159 PERSISTENT RESERVE IN parameter data for READ KEYS.....	277
160 PERSISTENT RESERVE IN parameter data for READ RESERVATION with no reservation held .....	278

161 PERSISTENT RESERVE IN parameter data for READ RESERVATION with reservation .....	278
162 Persistent reservation SCOPE field.....	279
163 Persistent reservation TYPE field .....	279
164 PERSISTENT RESERVE IN parameter data for REPORT CAPABILITIES.....	280
165 ALLOW COMMANDS field .....	281
166 Persistent Reservation Type Mask format .....	282
167 PERSISTENT RESERVE IN parameter data for READ FULL STATUS .....	283
168 PERSISTENT RESERVE IN full status descriptor format.....	284
169 PERSISTENT RESERVE OUT command.....	285
170 PERSISTENT RESERVE OUT service action codes .....	287
171 PERSISTENT RESERVE OUT parameter list.....	288
172 PERSISTENT RESERVE OUT specify initiator ports additional parameter data .....	289
173 PERSISTENT RESERVE OUT service actions and valid parameters (part 1 of 2).....	290
174 PERSISTENT RESERVE OUT command with REGISTER AND MOVE service action parameter list .....	291
175 READ ATTRIBUTE command .....	293
176 READ ATTRIBUTE service action codes.....	294
177 READ ATTRIBUTE with ATTRIBUTE VALUES service action parameter list format.....	295
178 READ ATTRIBUTE with ATTRIBUTE LIST service action parameter list format .....	295
179 READ ATTRIBUTE with VOLUME LIST service action parameter list format .....	296
180 READ ATTRIBUTE with PARTITION LIST service action parameter list format .....	296
181 READ BUFFER command .....	297
182 READ BUFFER MODE field .....	297
183 READ BUFFER header .....	298
184 READ BUFFER descriptor .....	299
185 OFFSET BOUNDARY field.....	299
186 Echo buffer descriptor.....	300
187 Error history BUFFER ID field .....	301
188 Summary of error history directory device server actions .....	302
189 Error history directory.....	302
190 EHS_RETRIEVED field .....	303
191 EHS_SOURCE field .....	303
192 Error history directory entry.....	304
193 READ MEDIA SERIAL NUMBER command.....	305
194 READ MEDIA SERIAL NUMBER parameter data format.....	305
195 RECEIVE COPY RESULTS command.....	306
196 RECEIVE COPY RESULTS service action codes .....	306
197 Parameter data for the COPY STATUS service action.....	307
198 COPY MANAGER STATUS field .....	308
199 COPY STATUS TRANSFER COUNT UNITS field.....	308
200 Parameter data for the RECEIVE DATA service action.....	309
201 Parameter data for the OPERATING PARAMETERS service action .....	310
202 Parameter data for the FAILED SEGMENT DETAILS service action.....	313
203 RECEIVE CREDENTIAL command.....	314
204 RECEIVE CREDENTIAL command unencrypted bytes format .....	315
205 CREDENTIAL REQUEST TYPE field.....	315
206 CbCS logical unit credential request descriptor format .....	316
207 CbCS logical unit and volume credential request descriptor format .....	316
208 CbCS credential format.....	317
209 Credential format values .....	317
210 CbCS capability descriptor format .....	318
211 DESIGNATION TYPE field .....	318
212 CbCS METHOD field .....	319
213 PERMISSIONS BIT MASK field format.....	319
214 Logical unit designation descriptor format .....	321

215 Volume designation descriptor format .....	321
216 RECEIVE DIAGNOSTIC RESULTS command.....	322
218 REPORT ALIASES parameter data.....	323
217 REPORT ALIASES command .....	323
219 REPORT IDENTIFYING INFORMATION command .....	324
220 INFORMATION TYPE field.....	325
221 REPORT IDENTIFYING INFORMATION parameter data.....	325
222 REPORT IDENTIFYING INFORMATION SUPPORTED parameter data .....	326
223 Identifying information descriptor .....	326
224 REPORT LUNS command.....	327
225 SELECT REPORT field .....	327
226 REPORT LUNS parameter data format.....	328
227 REPORT PRIORITY command .....	329
228 PRIORITY REPORTED field .....	329
229 REPORT PRIORITY parameter data format.....	330
230 Priority descriptor format.....	330
231 REPORT SUPPORTED OPERATION CODES command.....	331
232 REPORT SUPPORTED OPERATION CODES REPORTING OPTIONS field .....	332
233 All_commands parameter data .....	333
234 Command descriptor format .....	333
235 One_command parameter data .....	334
236 SUPPORT values.....	335
237 Command timeouts descriptor format.....	336
238 Command timeouts descriptor COMMAND SPECIFIC field usage in this standard .....	336
239 REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS command.....	337
240 REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS parameter data .....	338
241 REPORT TARGET PORT GROUPS command .....	339
242 REPORT TARGET PORT GROUPS parameter data format .....	340
243 Target port group descriptor format .....	340
244 ASYMMETRIC ACCESS STATE field .....	341
245 STATUS CODE field.....	342
246 Target port descriptor format .....	342
247 REPORT TIMESTAMP command .....	343
248 REPORT TIMESTAMP parameter data format.....	343
249 REQUEST SENSE command.....	344
250 DESC bit .....	344
251 SECURITY PROTOCOL IN command .....	347
252 SECURITY PROTOCOL field in SECURITY PROTOCOL IN command .....	347
253 SECURITY PROTOCOL OUT command .....	349
254 SECURITY PROTOCOL field in SECURITY PROTOCOL OUT command .....	349
255 SEND DIAGNOSTIC command.....	350
256 SELF-TEST CODE field .....	351
257 SET IDENTIFYING INFORMATION command .....	353
258 INFORMATION TYPE field.....	354
259 SET IDENTIFYING INFORMATION parameter list .....	354
260 SET PRIORITY command .....	355
261 I_T_L NEXUS TO SET field.....	356
262 SET PRIORITY parameter list format .....	357
263 SET TARGET PORT GROUPS command .....	358
264 SET TARGET PORT GROUPS parameter list format.....	359
265 Set target port group descriptor parameter list .....	360
266 ASYMMETRIC ACCESS STATE field .....	360
267 SET TIMESTAMP command .....	361
268 SET TIMESTAMP parameter data format .....	361

269 TEST UNIT READY command .....	362
270 Preferred TEST UNIT READY responses.....	362
271 WRITE ATTRIBUTE command.....	363
272 WRITE ATTRIBUTE parameter list format.....	364
273 WRITE BUFFER command .....	365
274 WRITE BUFFER MODE field .....	366
275 Application client error history parameter list format .....	370
276 ERROR TYPE field.....	371
277 ERROR LOCATION FORMAT field .....	372
278 Diagnostic page format .....	373
279 Diagnostic page codes.....	374
280 Protocol Specific diagnostic page .....	374
281 Supported Diagnostic Pages diagnostic page .....	375
282 Log page format.....	376
283 LOG SELECT PCR bit, SP bit, and DS bit meanings when parameter list length is not zero .....	377
284 Log parameter.....	378
285 Threshold met criteria (TMC) field .....	380
286 FORMAT AND LINKING field.....	380
287 Allowed LOG SELECT FORMAT AND LINKING field values .....	381
288 Parameter control byte values for data counter parameters .....	382
289 Parameter control byte values for list parameters.....	383
290 Log page codes .....	384
291 Application client log page .....	385
292 General usage application client parameter data .....	385
293 Parameter code field for buffer over-run/under-run counters.....	386
294 COUNT BASIS field.....	386
295 CAUSE field.....	387
296 Error Counter log page codes .....	387
297 Parameter codes for Error Counter log pages .....	387
298 Informational Exceptions log page.....	388
299 Informational exceptions parameter codes .....	388
300 Informational exceptions general parameter data.....	388
301 Non-medium error event parameter codes .....	390
302 Protocol Specific Port log page .....	391
303 Protocol specific port log parameter format .....	391
304 Self-Test Results log page .....	392
305 Self-test results log parameter format .....	393
306 SELF-TEST RESULTS field .....	394
307 Start-Stop Cycle Counter log page .....	395
308 Statistics and Performance log pages read and write commands .....	398
309 General Statistics and Performance log page .....	398
310 Statistics and Performance log parameter .....	399
311 Idle Time log parameter .....	401
312 Time Interval log parameter .....	402
313 Time interval descriptor.....	402
314 Force Unit Access Statistics and Performance log parameter .....	403
315 Group Statistics and Performance (n) log page.....	404
316 Group Statistics and Performance (n) subpage codes .....	405
317 Group n Statistics and Performance log parameter.....	406
318 Group n Force Unit Access Statistics and Performance log parameter.....	407
319 Supported Log Pages log page .....	408
320 Supported page descriptor.....	409
321 Supported Log Pages and Subpages log page .....	409
322 Supported page/subpage descriptor .....	409

323 Supported Subpages log page .....	410
324 Supported page/subpage descriptor .....	410
325 Temperature log page.....	411
326 MAM ATTRIBUTE format .....	413
327 MAM attribute FORMAT field .....	413
328 MAM attribute identifier range assignments.....	414
329 Device type attributes .....	415
330 DEVICE VENDOR/SERIAL NUMBER attribute format.....	416
331 MEDIUM USAGE HISTORY attribute format.....	417
332 PARTITION USAGE HISTORY attribute format .....	419
333 Medium type attributes.....	421
334 MEDIUM TYPE and MEDIUM TYPE INFORMATION attributes .....	421
335 Host type attributes .....	422
336 TEXT LOCALIZATION IDENTIFIER attribute values.....	423
337 Mode parameter list .....	424
338 Mode parameter header(6) .....	424
339 Mode parameter header(10) .....	425
340 General mode parameter block descriptor.....	426
341 Page_0 mode page format .....	427
342 Sub_page mode page format .....	427
343 Mode page codes and subpage codes .....	428
344 Control mode page .....	429
345 Task set type (TST) field .....	429
346 QUEUE ALGORITHM MODIFIER field .....	430
347 Queue error management (QERR) field .....	431
348 Unit attention interlocks control (UA_INTLCK_CTRL) field .....	432
349 AUTOLOAD MODE field .....	433
350 Control Extension mode page.....	434
351 Disconnect-Reconnect mode page.....	435
352 Data transfer disconnect control (DTDC) field .....	437
353 Extended mode page .....	438
354 Extended Device-Type Specific mode page .....	438
355 Informational Exceptions Control mode page .....	439
356 Method of reporting informational exceptions (MRIE) field.....	440
357 Power Condition mode page.....	442
358 Protocol Specific Logical Unit mode page .....	443
359 Page_0 format Protocol Specific Port mode page .....	444
360 Sub_page format Protocol Specific Port mode page .....	445
361 PROTOCOL IDENTIFIER values.....	446
362 Fibre Channel alias entry format codes .....	447
363 Fibre Channel world wide port name alias entry designation.....	447
364 Fibre Channel world wide port name with N_Port checking alias entry designation.....	448
365 RDMA alias entry format codes .....	448
366 RDMA target port identifier alias entry designation.....	449
367 InfiniBand global identifier with target port identifier checking alias entry designation.....	449
368 iSCSI alias entry format codes.....	450
369 iSCSI name alias entry designation .....	450
370 iSCSI name with binary IPv4 address alias entry designation .....	451
371 iSCSI name with IPname alias entry designation .....	452
372 iSCSI name with binary IPv6 address alias entry designation.....	453
373 Fibre Channel N_Port_Name EXTENDED COPY target descriptor format.....	454
374 Fibre Channel N_Port_ID EXTENDED COPY target descriptor format.....	455
375 Fibre Channel N_Port_ID with N_Port_Name checking target descriptor format .....	456
376 SCSI Parallel T_L EXTENDED COPY target descriptor format.....	457

377 IEEE 1394 EUI-64 EXTENDED COPY target descriptor format.....	458
378 RDMA EXTENDED COPY target descriptor format.....	459
379 iSCSI binary IPv4 address EXTENDED COPY target descriptor format .....	460
380 SAS serial SCSI protocol EXTENDED COPY target descriptor format .....	461
381 TransportID format.....	461
382 TransportID formats for specific SCSI transport protocols.....	462
383 Fibre Channel TransportID format .....	462
384 Parallel SCSI bus TransportID format.....	463
385 IEEE 1394 TransportID format.....	463
386 RDMA TransportID format .....	464
387 iSCSI TransportID formats .....	464
388 iSCSI initiator device TransportID format.....	464
389 iSCSI initiator port TransportID format .....	465
390 SAS Serial SCSI Protocol TransportID format .....	466
391 SECURITY PROTOCOL SPECIFIC field for SECURITY PROTOCOL IN protocol 00h.....	467
392 Supported security protocols SECURITY PROTOCOL IN parameter data .....	468
393 Certificate data SECURITY PROTOCOL IN parameter data.....	469
394 SECURITY PROTOCOL SPECIFIC field for the SA creation capabilities SECURITY PROTOCOL IN command ..	470
395 Supported device server capabilities formats parameter data .....	471
396 IKEv2-SCSI device server capabilities parameter data .....	471
397 SECURITY PROTOCOL SPECIFIC field for the IKEv2-SCSI SECURITY PROTOCOL IN command .....	472
398 SECURITY PROTOCOL SPECIFIC field for the IKEv2-SCSI SECURITY PROTOCOL OUT command .....	473
399 IKEv2-SCSI SECURITY PROTOCOL OUT and SECURITY PROTOCOL IN parameter data.....	474
400 IKEv2-SCSI header checking of SAs .....	476
401 NEXT PAYLOAD field .....	477
402 MESSAGE ID field .....	478
403 Next payload values in SECURITY PROTOCOL OUT/IN parameter data .....	479
404 IKEv2-SCSI payload format .....	482
405 Key Exchange payload format .....	483
406 Identification payload format .....	484
407 ID TYPE field .....	484
408 Certificate Request payload and Certificate payload format.....	485
409 CERTIFICATE ENCODING field .....	485
410 Authentication payload format .....	486
411 Nonce payload format.....	489
412 Notify payload format .....	490
413 Delete payload format.....	491
414 Encrypted payload format.....	493
415 Plaintext format for Encrypted payload CIPHERTEXT field .....	495
416 IKEv2-SCSI SA Creation Capabilities payload format .....	498
417 IKEv2-SCSI SA Cryptographic Algorithms payload format.....	499
418 IKEv2-SCSI SAUT Cryptographic Algorithms payload format .....	501
419 IKEv2-SCSI Timeout Values payload format .....	502
420 IKEv2-SCSI cryptographic algorithm descriptor format.....	503
421 ALGORITHM TYPE field.....	504
422 ENCR IKEv2-SCSI cryptographic algorithm descriptor format .....	505
423 ENCR ALGORITHM IDENTIFIER field .....	506
424 PRF IKEv2-SCSI cryptographic algorithm descriptor format .....	507
425 PRF ALGORITHM IDENTIFIER field .....	508
426 INTEG IKEv2-SCSI cryptographic algorithm descriptor format.....	509
427 INTEG ALGORITHM IDENTIFIER field .....	509
428 D-H IKEv2-SCSI cryptographic algorithm descriptor format .....	510
429 D-H ALGORITHM IDENTIFIER field .....	511
430 SA_AUTH_OUT and SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor format .....	512



431 SA_AUTH_OUT and SA_AUTH_IN ALGORITHM IDENTIFIER field .....	513
432 IKEv2-SCSI command ordering processing requirements on a single I_T_L nexus .....	515
433 IKEv2-SCSI parameter error categories .....	517
434 IKEv2 Notify payload error translations for IKEv2-SCSI .....	519
435 SECURITY PROTOCOL SPECIFIC field for the CbCS SECURITY PROTOCOL IN command .....	520
436 Supported CbCS SECURITY PROTOCOL IN Pages CbCS page format .....	521
437 Supported CbCS SECURITY PROTOCOL OUT Pages CbCS page format .....	522
438 Unchangeable CbCS Parameters CbCS page format .....	523
439 KEYS SUPPORT field .....	524
440 MIN CBCS METHOD SUP field .....	524
441 Security Token CbCS page format .....	525
442 Current CbCS Parameters CbCS page format .....	526
443 Set Master Key – Seed Exchange CbCS page format .....	527
444 SECURITY PROTOCOL SPECIFIC field for the CbCS SECURITY PROTOCOL OUT command .....	529
445 Set Policy Access Tag CbCS page format .....	530
446 Set Minimum CbCS Method CbCS page format .....	530
447 Minimum CbCS Method CbCS Parameter set .....	531
448 Invalidate Key CbCS page format .....	532
449 Set Key CbCS page format .....	533
450 Set Master Key – Seed Exchange CbCS page format .....	534
451 Set Master Key – Change Master Key CbCS page format .....	535
452 Vital product data page codes .....	537
453 ASCII Information VPD page .....	538
454 Device Identification VPD page .....	539
455 Designation descriptor .....	540
456 ASSOCIATION field .....	540
457 DESIGNATOR TYPE field .....	541
458 Vendor specific DESIGNATOR field format .....	543
459 T10 vendor ID based DESIGNATOR field format .....	543
460 EUI-64 based designator lengths .....	544
461 EUI-64 DESIGNATOR field format .....	544
462 EUI-64 based 12-byte DESIGNATOR field format .....	545
463 EUI-64 based 16-byte DESIGNATOR field format .....	545
464 NAA DESIGNATOR field format .....	546
465 Name Address Authority (NAA) field .....	546
466 NAA IEEE Extended DESIGNATOR field format .....	546
467 NAA Locally Assigned DESIGNATOR field format .....	547
468 NAA IEEE Registered DESIGNATOR field format .....	547
469 NAA IEEE Registered Extended DESIGNATOR field format .....	548
470 Relative target port DESIGNATOR field format .....	548
471 RELATIVE TARGET PORT IDENTIFIER field .....	549
472 Target port group DESIGNATOR field format .....	549
473 Logical unit group DESIGNATOR field format .....	549
474 MD5 logical unit DESIGNATOR field format .....	550
475 MD5 logical unit identifier example available data .....	550
476 Example MD5 input for computation of a logical unit identifier .....	551
477 SCSI name string DESIGNATOR field format .....	551
478 Extended INQUIRY Data VPD page .....	552
479 Supported protection type (SPT) field .....	552
480 Management Network Addresses VPD page .....	554
481 Network service descriptor format .....	555
482 SERVICE TYPE field .....	555
483 Mode Page Policy VPD page .....	556
484 Mode page policy descriptor .....	556

485 MODE PAGE POLICY field .....	557
486 SCSI Ports VPD page.....	558
487 SCSI port designation descriptor .....	559
488 RELATIVE PORT IDENTIFIER field.....	559
489 Target port descriptor.....	560
490 Protocol Specific Logical Unit Information VPD page .....	561
491 Logical unit information descriptor .....	561
492 Protocol Specific Port Information VPD page .....	562
493 Port information descriptor .....	563
494 Software Interface Identification VPD page .....	564
495 Software interface identifier format .....	564
496 Supported VPD Pages VPD page .....	565
497 Unit Serial Number VPD page .....	565
498 Well known logical unit numbers.....	566
499 Commands for the REPORT LUNS well known logical unit .....	566
500 Commands for the ACCESS CONTROLS well known logical unit .....	567
501 ACCESS CONTROL OUT management identifier key requirements .....	570
502 ACCESS CONTROL IN management identifier key requirements .....	571
503 Mandatory access controls resources .....	581
504 Optional access controls resources .....	582
505 ACCESS IDENTIFIER TYPE field .....	582
506 AccessID access identifier format.....	582
507 ACCESS CONTROL IN service actions .....	583
508 ACCESS CONTROL IN command with REPORT ACL service action .....	584
509 ACCESS CONTROL IN with REPORT ACL parameter data format .....	585
510 ACL data page codes .....	585
511 Granted ACL data page format.....	586
512 Granted ACL data page LUACD descriptor format.....	587
513 ACCESS MODE field .....	587
514 Granted All ACL data page format.....	588
515 Proxy Tokens ACL data page format.....	589
516 Proxy token descriptor format .....	589
517 ACCESS CONTROL IN command with REPORT LU DESCRIPTORS service action .....	590
518 ACCESS CONTROL IN with REPORT LU DESCRIPTORS parameter data format.....	591
519 SUPPORTED LUN MASK FORMAT field format .....	592
520 Logical Unit descriptor format .....	593
521 ACCESS CONTROL IN command with REPORT ACCESS CONTROLS LOG service action.....	595
522 CDB LOG PORTION field.....	596
523 ACCESS CONTROL IN with REPORT ACCESS CONTROLS LOG parameter data format.....	596
524 Parameter data LOG PORTION field .....	597
525 Key Overrides access controls log portion page format.....	597
526 Invalid Keys access controls log portion page format .....	598
527 ACL LUN Conflicts access controls log portion page format .....	599
528 ACCESS CONTROL IN command with REPORT OVERRIDE LOCKOUT TIMER service action.....	600
529 ACCESS CONTROL IN with REPORT OVERRIDE LOCKOUT TIMER parameter data .....	600
530 ACCESS CONTROL IN command with REQUEST PROXY TOKEN service action.....	601
531 ACCESS CONTROL IN with REQUEST PROXY TOKEN parameter data .....	602
532 ACCESS CONTROL OUT service actions .....	603
533 ACCESS CONTROL OUT command format .....	604
534 ACCESS CONTROL OUT with MANAGE ACL parameter data format.....	605
535 ACE page codes .....	606
536 Grant/Revoke ACE page format .....	607
537 ACE page LUACD descriptor format .....	608
538 Access Coordinator Grant/Revoke ACE page actions.....	609

539 Grant All ACE page format .....	609
540 Revoke Proxy Token ACE page format .....	610
541 Revoke All Proxy Tokens ACE page format .....	611
542 ACCESS CONTROL OUT with DISABLE ACCESS CONTROLS parameter data format .....	611
543 ACCESS CONTROL OUT with ACCESS ID ENROLL parameter data format .....	612
544 ACCESS CONTROL OUT with CLEAR ACCESS CONTROLS LOG parameter data format.....	614
545 CLEAR ACCESS CONTROLS LOG LOG PORTION field .....	614
546 ACCESS CONTROL OUT with MANAGE OVERRIDE LOCKOUT TIMER parameter data format .....	615
547 ACCESS CONTROL OUT with OVERRIDE MGMT ID KEY parameter data format.....	616
548 ACCESS CONTROL OUT with REVOKE PROXY TOKEN parameter data format .....	617
549 ACCESS CONTROL OUT with REVOKE ALL PROXY TOKENS parameter data format .....	618
550 ACCESS CONTROL OUT with ASSIGN PROXY LUN parameter data format.....	619
551 ACCESS CONTROL OUT with RELEASE PROXY LUN parameter data format.....	620
552 Commands for the TARGET LOG PAGES well known logical unit .....	621
553 Commands for the SECURITY PROTOCOL well known logical unit.....	621
554 Commands for the MANAGEMENT PROTOCOL well known logical unit .....	622
555 Commands for a security manager .....	623
A.1 This standard to SPC-2 terminology mapping .....	624
B.1 PERSISTENT RESERVE OUT command features.....	625
C.1 IKE payload names shorthand .....	629
D.1 ASC and ASCQ assignments.....	630
D.2 Operation codes .....	647
D.3 Additional operation codes for devices with the EncServ bit set to one .....	653
D.4 MAINTENANCE (IN) and MAINTENANCE (OUT) service actions .....	654
D.5 SERVICE ACTION IN(12) and SERVICE ACTION OUT(12) service actions .....	655
D.6 SERVICE ACTION IN(16) and SERVICE ACTION OUT(16) service actions .....	655
D.7 Variable Length CDB Service Action Code Ranges.....	656
D.8 Variable Length CDB Service Action Codes Used by All Device Types .....	656
D.9 Diagnostic page codes .....	657
D.10 Log page codes .....	658
D.11 Transport protocol specific log page codes .....	660
D.12 Mode page codes .....	661
D.13 Transport protocol specific mode page codes .....	663
D.14 VPD page codes .....	664
D.15 Transport protocol specific VPD page codes .....	665
D.16 Version descriptor assignments .....	666
D.17 Standard code value guidelines .....	677
D.18 Revision code value guidelines .....	680
D.19 IEEE binary identifiers assigned by T10.....	681
E.1 T10 vendor identification list .....	682

## Figures

	Page
1 SCSI document relationships .....	1
2 Example state diagram .....	29
3 Example class association relationships .....	32
4 Example class aggregation relationships .....	33
5 Example class generalization relationships .....	34
6 Example class dependency relationships .....	35
7 Device server interpretation of PREEMPT service action .....	101
8 Primary target port group example .....	106
9 Power condition state machine .....	114
10 SA relationships .....	124
11 IKEv2-SCSI Device Server Capabilities step .....	135
12 IKEv2-SCSI Key Exchange step .....	135
13 IKEv2-SCSI Authentication step .....	136
14 IKEv2-SCSI Delete operation .....	137
15 CbCS overview class diagram .....	159
16 ACL Structure .....	569

## Foreword

This foreword is not part of American National Standard INCITS.\*\*\*:200x.

The SCSI command set is designed to provide efficient peer-to-peer operation of SCSI devices (e.g., disks, tapes, printers) by an operating system. The SCSI command set assumes an underlying command-response protocol.

The SCSI command set provides multiple operating systems concurrent control over one or more SCSI devices. However, proper coordination of activities between the multiple operating systems is critical to avoid data corruption. Commands that assist with coordination between multiple operating systems are described in this standard. However, details of the coordination are beyond the scope of the SCSI command set.

This standard defines the device model for all SCSI devices. This standard defines the SCSI commands that are basic to every device model and the SCSI commands that may apply to any device model.

With any technical document there may arise questions of interpretation as new products are implemented. INCITS has established procedures to issue technical opinions concerning the standards developed by INCITS. These procedures may result in SCSI Technical Information Bulletins being published by INCITS.

These Bulletins, while reflecting the opinion of the Technical Committee that developed the standard, are intended solely as supplementary information to other users of the standard. This standard, ANSI INCITS.\*\*\*:200x, as approved through the publication and voting procedures of the American National Standards Institute, is not altered by these bulletins. Any subsequent revision to this standard may or may not reflect the contents of these Technical Information Bulletins.

Current INCITS practice is to make Technical Information Bulletins available through:

INCITS Online Store	<a href="http://www.techstreet.com/incits.html">http://www.techstreet.com/incits.html</a>
managed by Techstreet	Telephone: 1-734-302-7801 or
1327 Jones Drive	1-800-699-9277
Ann Arbor, MI 48105	Facsimile: 1-734-302-7811

or

Global Engineering	<a href="http://global.ihs.com/">http://global.ihs.com/</a>
15 Inverness Way East	Telephone: 1-303-792-2181 or
Englewood, CO 80112-5704	1-800-854-7179
	Facsimile: 1-303-792-2192

Requests for interpretation, suggestions for improvement and addenda, or defect reports are welcome. They should be sent to the INCITS Secretariat, National Committee for Information Technology Standards, Information Technology Institute, 1250 Eye Street, NW, Suite 200, Washington, DC 20005-3922.

This standard was processed and approved for submittal to ANSI by the InterNational Committee for Information Technology Standards (INCITS). Committee approval of the standard does not necessarily imply that all committee members voted for approval. At the time of it approved this standard, INCITS had the following members:

<<Insert INCITS member list>>

Technical Committee T10 on SCSI Storage Interfaces, which developed and reviewed this standard, had the following members:

John B. Lohmeyer, Chair  
George O. Penokie, Vice-Chair  
Ralph O. Weber, Secretary

<<Insert T10 member list>>

## Introduction

The SCSI Primary Commands - 4 (SPC-4) standard is divided into the following clauses and annexes:

Clause 1 is the scope.

Clause 2 enumerates the normative references that apply to this standard.

Clause 3 describes the definitions, symbols, and abbreviations used in this standard.

Clause 4 describes the conceptual relationship between this document and the SCSI-3 Architecture Model.

Clause 5 describes the command model for all SCSI devices.

Clause 6 defines the commands that may be implemented by any SCSI device.

Clause 7 defines the parameter data formats that may be implemented by any SCSI device.

Clause 8 defines the well known logical units that may be implemented by any SCSI device.

Annex A identifies differences between the terminology used in this standard and previous versions of this standard. (informative)

Annex B describes the PERSISTENT RESERVE OUT command features necessary to replace the reserve/release management method and provides guidance on how to perform a third party reservation using persistent reservations. (informative)

Annex C identifies the differences between security protocols defined by other standards (e.g., IKEv2 (see RFC 4306)) and the equivalent protocols defined by this standard (e.g., the IKEv2-SCSI SA creation protocol). (informative)

Annex D lists code values in numeric order. (informative)

Annex E lists assigned vendor identifiers. (informative)

The annexes provide information to assist with implementation of this standard. The information in the annexes applies to all the SCSI command standards. See 3.1.27 for more information about other SCSI command standards.

**AMERICAN NATIONAL STANDARD****INCITS.\*\*\*:200x****American National Standard  
for Information Technology -****SCSI Primary Commands - 4 (SPC-4)****1 Scope**

The SCSI family of standards provides for many different types of SCSI devices (e.g., disks, tapes, printers, scanners). This standard defines a device model that is applicable to all SCSI devices. Other SCSI command standards (see 3.1.27) expand on the general SCSI device model in ways appropriate to specific types of SCSI devices.

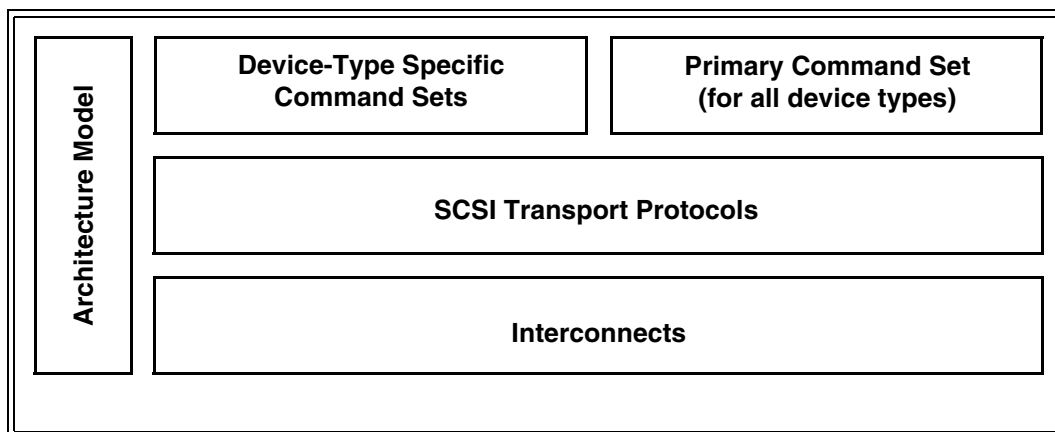
The set of SCSI standards specifies the interfaces, functions, and operations necessary to ensure interoperability between conforming SCSI implementations. This standard is a functional description. Conforming implementations may employ any design technique that does not violate interoperability.

This standard defines the SCSI commands that are mandatory and optional for all SCSI devices. Support for any feature defined in this standard is optional unless otherwise stated. This standard also defines the SCSI commands that may apply to any device model.

The following commands, parameter data, and features defined in previous versions of this standard are made obsolete by this standard:

- a) Code value 10b (i.e., Per initiator port) in the MODE PAGE POLICY field in the mode page policy descriptor in the Mode Page Policy VPD page;
- b) The removable medium devices with an attached medium changer model, MCHNGR bit in the standard INQUIRY data, the MOVE MEDIUM ATTACHED command in disks and tapes, and the READ ELEMENT STATUS ATTACHED command in disks and tapes; and
- c) Linked commands.

Figure 1 shows the relationship of this standard to the other standards and related projects in the SCSI family of standards as of the publication of this standard.



**Figure 1 — SCSI document relationships**



Figure 1 is intended to show the general relationship of the documents to one another. Figure 1 is not intended to imply a relationship such as a hierarchy, protocol stack, or system architecture. It indicates the applicability of a standard to the implementation of a given transport.

At the time this standard was generated, examples of the SCSI general structure included:

Interconnects:

Fibre Channel Arbitrated Loop - 2	FC-AL-2	[ISO/IEC 14165-122] [ANSI INCITS 332-1999] [ANSI INCITS 332/AM1-2003] [ANSI INCITS 332/AM2-2006]
Fibre Channel Physical Interfaces	FC-PI	[ISO/IEC 14165-115] [ANSI INCITS 352-2002]
Fibre Channel Physical Interfaces - 2	FC-PI-2	[ISO/IEC 14165-142] [ANSI INCITS 404-2006]
Fibre Channel Physical Interfaces - 3	FC-PI-3	[ISO/IEC 14165-143] [T11/1625-D]
Fibre Channel Physical Interfaces - 4	FC-PI-4	[ISO/IEC 14165-144] [T11/1647-D]
Fibre Channel - 10 Gigabit	10GFC	[ISO/IEC 14165-116] [ANSI INCITS 364-2003] [ANSI INCITS 364/AM1-2007]
Fibre Channel Framing and Signaling Interface	FC-FS	[ISO/IEC 14165-251] [ANSI INCITS 373-2003]
Fibre Channel Framing and Signaling Interface - 2	FC-FS-2	[ISO/IEC 14165-252] [ANSI INCITS 424-2007] [ANSI INCITS 424/AM1-2007]
Fibre Channel Framing and Signaling Interface - 3	FC-FS-3	[ISO/IEC 14165-253] [T11/1861-D]
Fibre Channel Link Services	FC-LS	[ANSI INCITS 433-2007]
Fibre Channel Link Services - 2	FC-LS-2	[T11/2103-D]
Fibre Channel Security Protocols	FC-SP	[ISO/IEC 14165-431] [ANSI INCITS 426-2007]
High Performance Serial Bus		[ANSI IEEE 1394-1995]
High Performance Serial Bus (supplement to ANSI/IEEE 1394-1995)		[ANSI IEEE 1394a-2000]
SCSI Parallel Interface - 2	SPI-2	[ISO/IEC 14776-112] [ANSI INCITS 302-1999]
SCSI Parallel Interface - 3	SPI-3	[ISO/IEC 14776-113] [ANSI INCITS 336-2000]
SCSI Parallel Interface - 4	SPI-4	[ISO/IEC 14776-114] [ANSI INCITS 362-2002]
SCSI Parallel Interface - 5	SPI-5	[ISO/IEC 14776-115] [ANSI INCITS 367-2003]
Serial Storage Architecture Physical Layer 1	SSA-PH	[ANSI INCITS 293-1996]
Serial Storage Architecture Physical Layer 2	SSA-PH-2	[ANSI INCITS 307-1998]
Serial Attached SCSI	SAS	[ISO/IEC 14776-150] [ANSI INCITS 376-2003]
Serial Attached SCSI - 1.1	SAS-1.1	[ISO/IEC 14776-151] [ANSI INCITS 417-2006]
Serial Attached SCSI - 2	SAS-2	[ISO/IEC 14776-152] [T10/1760-D]

## SCSI Transport Protocols:

Automation/Drive Interface - Transport Protocol	ADT	[ANSI INCITS 406-2005]
Automation/Drive Interface - Transport Protocol - 2	ADT-2	[ISO/IEC 14776-192] [T10/1742-D]
Serial Storage Architecture Transport Layer 1	SSA-TL-1	[ANSI INCITS 295-1996]
Serial Storage Architecture Transport Layer 2	SSA-TL-2	[ANSI INCITS 308-1998]
SCSI-3 Fibre Channel Protocol	FCP	[ISO/IEC 14776-221] [ANSI INCITS 269-1996]
Fibre Channel Protocol for SCSI - 2	FCP-2	[ISO/IEC 14776-222] [ANSI INCITS 350-2003]
Fibre Channel Protocol for SCSI - 3	FCP-3	[ISO/IEC 14776-223] [ANSI INCITS 416-2006]
Fibre Channel Protocol for SCSI - 4	FCP-4	[ISO/IEC 14776-223] [T10/1828-D]
Serial Bus Protocol - 2	SBP-2	[ISO/IEC 14776-232] [ANSI INCITS 325-1998]
Serial Bus Protocol - 3	SBP-3	[ANSI INCITS 375-2004]
Serial Storage Architecture SCSI-3 Protocol	SSA-S3P	[ANSI INCITS 309-1998]
SCSI RDMA Protocol	SRP	[ISO/IEC 14776-241] [ANSI INCITS 365-2002]
USB Queuing Transport	UAS	[ISO/IEC 14776-251] [T10/2095-D]

## Primary Command Set:

SCSI-3 Primary Commands	SPC	[ANSI INCITS 301-1997]
SCSI Primary Commands - 2	SPC-2	[ISO/IEC 14776-452] [ANSI INCITS 351-2001]
SCSI Primary Commands - 3	SPC-3	[ISO/IEC 14776-453] [ANSI INCITS 408-2005]
SCSI Primary Commands - 4	SPC-4	[ISO/IEC 14776-454] [T10/1731-D]

## Device-Type Specific Command Sets:

SCSI-3 Block Commands	SBC	[ISO/IEC 14776-321] [ANSI INCITS 306-1998]
SCSI Block Commands - 2	SBC-2	[ISO/IEC 14776-322] [ANSI INCITS 405-2005]
SCSI Block Commands - 3	SBC-3	[ISO/IEC 14776-323] [T10/1799-D]
SCSI-3 Stream Commands	SSC	[ISO/IEC 14776-331] [ANSI INCITS 335-2000]
SCSI Stream Commands - 2	SSC-2	[ISO/IEC 14776-332] [ANSI INCITS 380-2003]
SCSI Stream Commands - 3	SSC-3	[ISO/IEC 14776-333] [T10/1611-D]
SCSI-3 Medium Changer Commands	SMC	[ISO/IEC 14776-351] [ANSI INCITS 314-1998]
SCSI Media Changer Commands - 2	SMC-2	[ISO/IEC 14776-352] [ANSI INCITS 382-2004]
SCSI Media Changer Commands - 3	SMC-3	[ISO/IEC 14776-353] [T10/1730-D]
SCSI-3 Multimedia Command Set	MMC	[ANSI INCITS 304-1997]
SCSI Multimedia Command Set - 2	MMC-2	[ISO/IEC 14776-362] [ANSI INCITS 333-2000]

SCSI Multimedia Command Set - 3	MMC-3	[ISO/IEC 14776-363] [ANSI INCITS 360-2002]
SCSI Multimedia Command Set - 4	MMC-4	[ISO/IEC 14776-364] [ANSI INCITS 401-2005]
SCSI Multimedia Command Set - 5	MMC-5	[ISO/IEC 14776-365] [ANSI INCITS 430-2007]
SCSI Multimedia Command Set - 6	MMC-6	[ISO/IEC 14776-366] [T10/1836-D]
SCSI Controller Commands - 2	SCC-2	[ISO/IEC 14776-342] [ANSI INCITS 318-1998]
SCSI Reduced Block Commands	RBC	[ISO/IEC 14776-326] [ANSI INCITS 330-2000] [ANSI INCITS 330-2003/AM1]
SCSI-3 Enclosure Services Commands	SES	[ISO/IEC 14776-371] [ANSI INCITS 305-1998] [ANSI INCITS 305-2000/AM1]
SCSI Enclosure Services Commands - 2	SES-2	[ISO/IEC 14776-372] [T10/1559-D]
SCSI Specification for Optical Card Reader/Writer Object-based Storage Device Commands	OCRW OSD	[ISO/IEC 14776-381] [ISO/IEC 14776-391] [ANSI INCITS 400-2004]
Object-based Storage Device Commands - 2	OSD-2	[ISO/IEC 14776-392] [T10/1729-D]
SCSI Bridge Controller Commands	BCC	[ISO/IEC 14776-511] [T10/1528-D]
Automation/Drive Interface - Commands	ADC	[ISO/IEC 14776-356] [ANSI INCITS 403-2005]
Automation/Drive Interface - Commands - 2	ADC-2	[ISO/IEC 14776-357] [T10/1741-D]
Automation/Drive Interface - Commands - 3	ADC-3	[ISO/IEC 14776-358] [T10/1895-D]
Translation Protocols:		
SCSI / ATA Translation	SAT	[ISO/IEC 14776-921] [ANSI INCITS 431-2007]
SCSI / ATA Translation - 2	SAT-2	[ISO/IEC 14776-922] [T10/1826-D]
Architecture Model:		
SCSI-3 Architecture Model	SAM	[ISO/IEC 14776-411] [ANSI INCITS 270-1996]
SCSI Architecture Model - 2	SAM-2	[ISO/IEC 14776-412] [ANSI INCITS 366-2003]
SCSI Architecture Model - 3	SAM-3	[ISO/IEC 14776-413] [ANSI INCITS 402-2005]
SCSI Architecture Model - 4	SAM-4	[ISO/IEC 14776-414] [T10/1683-D]
SCSI Architecture Model - 5	SAM-5	[ISO/IEC 14776-415] [T10/2104-D]

The term SCSI is used to refer to the family of standards described in this clause.

## 2 Normative references

### 2.1 General

The following standards contain provisions that, by reference in the text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed below.

### 2.2 Approved references

Copies of the following documents may be obtained from ANSI, an ISO member organization:

- a) Approved ANSI standards;
- b) approved international and regional standards (ISO and IEC); and
- c) approved foreign standards (including JIS and DIN).

For further information, contact the ANSI Customer Service Department:

Phone +1 212-642-4900  
Fax: +1 212-302-1286  
Web: <http://www.ansi.org>  
E-mail: [ansionline@ansi.org](mailto:ansionline@ansi.org)

or the InterNational Committee for Information Technology Standards (INCITS):

Phone +1 202-626-5738  
Web: <http://www.incits.org>  
E-mail: [incits@itic.org](mailto:incits@itic.org)

ISO/IEC 14776-412, *SCSI Architecture Model - 2 (SAM-2)* [ANSI INCITS 366-2003]

ISO/IEC 14776-452, *SCSI Primary Commands - 2 (SPC-2)* [ANSI INCITS 351-2001]

ISO/IEC 14776-113, *SCSI Parallel Interface - 5 (SPI-5)* [ANSI INCITS 367-2003]

ISO/IEC 14776-222, *Fibre Channel Protocol for SCSI - 2 (FCP-2)* [ANSI INCITS 350:2003]

ISO/IEC 14776-241, *SCSI RDMA Protocol (SRP)* [ANSI INCITS 365-2002]

ISO/IEC 14776-150, *Serial Attached SCSI (SAS)* [ANSI INCITS 376-2003]

ISO/IEC 14776-921, *SCSI / ATA Translation (SAT)* [ANSI INCITS 431-2007]

ISO/IEC 14776-413, *SCSI Architecture Model - 3 (SAM-3)* [ANSI INCITS 402-2005]

ISO/IEC 14776-322, *SCSI Block Commands - 2 (SBC-2)* [ANSI INCITS 405-2005]

ISO/IEC 14776-332, *SCSI Stream Commands - 2 (SSC-2)* [ANSI INCITS 380-2003]

ISO/IEC 14776-352, *SCSI Media Changer Commands - 2, (SMC-2)* [ANSI INCITS 382:2004]

ANSI INCITS 309-1998, *Serial Storage Architecture SCSI-3 Protocol (SSA-S3P)*

ANSI INCITS 375-2004, *Serial Bus Protocol - 3 (SBP-3)*

ANSI INCITS 406-2005, *Automation/Drive Interface - Transport Protocol (ADT)*

ISO/IEC 14165-251, *Fibre Channel Framing and Signaling Interface (FC-FS) [ANSI INCITS 373-2003]*

ISO/IEC 14165-431, *Fibre Channel Security Protocols (FC-SP) [ANSI INCITS 426-2007]*

ISO/IEC 24739 (all parts), *Information technology – (all parts) AT Attachment with Packet Interface - 7 (AT/ATAPI-7)*

IEC 60027:2000, *Letter symbols to be used in electrical technology - Part 2: Telecommunications and electronics*

ANSI/IEEE 1394a-2000, *High Performance Serial Bus (supplement to ANSI/IEEE 1394-1995)*

ANSI/IEEE 1619.1-20xx, *Standard for Authenticated Encryption with Length Expansion for Storage Devices*

ANSI INCITS 4-1986 (R2002), *Information Systems - Coded Character Sets - 7-Bit American National Standard Code for Information Interchange (7-Bit ASCII)*

ISO/IEC 13213, *Information technology - Microprocessor systems - Control and Status Registers Architecture for microcomputer buses [ANSI/IEEE 1212, 2001 Edition]*

ISO/IEC 646:1991, *Information technology - ISO 7-bit coded character set for information interchange (third edition).*

ISO/IEC 8859-1:1998, *Information technology - 8-bit single-byte coded graphic character sets - Part 1: Latin alphabet No. 1*

ISO/IEC 8859-2:1999, *Information technology - 8-bit single-byte coded graphic character sets - Part 2: Latin alphabet No. 2*

ISO/IEC 8859-3:1999, *Information technology - 8-bit single-byte coded graphic character sets - Part 3: Latin alphabet No. 3*

ISO/IEC 8859-4:1998, *Information technology - 8-bit single-byte coded graphic character sets - Part 4: Latin alphabet No. 4*

ISO/IEC 8859-5:1999, *Information technology - 8-bit single-byte coded graphic character sets - Part 5: Latin/Cyrillic alphabet*

ISO/IEC 8859-6:1999, *Information technology - 8-bit single-byte coded graphic character sets - Part 6: Latin/Arabic alphabet*

ISO/IEC 8859-7:1987, *Information processing - 8-bit single-byte coded graphic character sets - Part 7: Latin/Greek alphabet*

ISO/IEC 8859-8:1999, *Information technology - 8-bit single-byte coded graphic character sets - Part 8: Latin/Hebrew alphabet*

ISO/IEC 8859-9:1999, *Information technology - 8-bit single-byte coded graphic character sets - Part 9: Latin alphabet No. 5*

ISO/IEC 8859-10:1998, *Information technology - 8-bit single-byte coded graphic character sets - Part 10: Latin alphabet No. 6*

ISO/IEC 10646:2003, *Information technology - Universal Multiple-Octet Coded Character Set (UCS)*

ISO/IEC 9594-8:2001, *Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks [ITU-T Recommendation X.509:2000 – <http://www.itu.int/ITU-T/>]*

ISO/IEC 9594-8:2001/Cor 1:2002, *Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks*

ISO/IEC 9594-8:2001/Cor 2:2002, *Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks*

ISO/IEC 9594-8:2001/Cor 3:2005, *Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks*

ISO/IEC 8824-1:2002, *Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation*

ISO/IEC 8824-1:2002/Amd 1:2004, *Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation*

ISO/IEC 8824-1:2002/Amd 2:2005, *Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation*

ISO/IEC 8824-2:2002, *Information technology - Abstract Syntax Notation One (ASN.1): Information object specification*

ISO/IEC 8824-2:2002/Amd 1:2004, *Information technology - Abstract Syntax Notation One (ASN.1): Information object specification*

ISO/IEC 8824-3:2002, *Information technology - Abstract Syntax Notation One (ASN.1): Constraint specification*

ISO/IEC 8824-4:2002, *Information technology - Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications*

ISO/IEC 9834-1:2005, *Information technology - Open Systems Interconnection - Procedures for the operation of OSI Registration Authorities: General procedures and top arcs of the ASN.1 Object Identifier tree*

ANSI/IEEE 1667, *Standard Protocol for Authentication in Host Attachments of Transient Storage Devices*

## 2.3 References under development

At the time of publication, the following referenced standards were still under development. For information on the current status of the document, or regarding availability, contact the relevant standards body or other organization as indicated.

ISO/IEC 14776-414, *SCSI Architecture Model - 4 (SAM-4) [T10/1683-D]*

ISO/IEC 14776-323, *SCSI Block Commands - 3 (SBC-3) [T10/1799-D]*

ISO/IEC 14776-333, *SCSI Stream Commands - 3 (SSC-3)* [T10/1611-D]

ISO/IEC 14776-353, *SCSI Media Changer Commands - 3 (SMC-3)* [T10/1730-D]

ISO/IEC 14776-358, *Automation/Drive Interface - Commands - 3* [T10/1895-D]

ISO/IEC 14776-922, *SCSI / ATA Translation - 2 (SAT-2)* [T10/1826-D]

NIST SP (Special Publication) 800-38D, *Recommendation for Block Cipher Modes of Operation: Galois/Counter (GCM) Mode for Confidentiality and Authentication*

## 2.4 NIST References

Copies of the following approved FIPS standards may be obtained through the National Institute of Standards and Technology (NIST) at <http://csrc.nist.gov/publications/fips/index.html>.

FIPS 140-2, *Annex C: Approved Random Number Generators*

FIPS 180-2 with Change Notice 1 dated February 25, 2004, *Secure Hash Standard*

FIPS 198a, *The Keyed-Hash Message Authentication Code (HMAC)*

## 2.5 IETF References

Copies of the following approved IETF standards may be obtained through the Internet Engineering Task Force (IETF) at [www.ietf.org](http://www.ietf.org).

RFC 791, *Internet Protocol - DARPA Internet Program - Protocol Specification*

RFC 793, *Transmission Control Protocol - DARPA Internet Program - Protocol Specification*

RFC 1035, *Domain Names - Implementation and Specification*

RFC 1321, *The MD5 Message-Digest Algorithm*

RFC 1591, *Domain Name System Structure and Delegation*

RFC 2104, *HMAC: Keyed-Hashing for Message Authentication*

RFC 2279, *UTF-8, a transformation format of ISO 10646*

RFC 2373, *IP Version 6 Addressing Architecture*

RFC 2396, *Uniform Resource Identifiers (URI): Generic Syntax*

RFC 2410, *The NULL Encryption Algorithm and Its Use With IPsec*

RFC 2437, *PKCS #1: RSA Cryptography Specifications Version 2.0*

RFC 2616, *Hypertext Transfer Protocol - HTTP/1.1*

RFC 2753, *A Framework for Policy-based Admission Control*

RFC 3280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*

RFC 3281, *An Internet Attribute Certificate Profile for Authorization*

RFC 3305, *Report from the Joint W3C/IETF URI Planning Interest Group: Uniform Resource Identifiers (URIs), URLs, and Uniform Resource Names (URNs): Clarifications and Recommendations*

RFC 3447, *Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1*

RFC 3526, *More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)*

RFC 3552, *Guidelines for Writing RFC Text on Security Considerations*

RFC 3566, *The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec*

RFC 3602, *The AES-CBC Cipher Algorithm and Its Use with IPsec*

RFC 3720, *Internet Small Computer Systems Interface (iSCSI)*

RFC 3766, *Determining Strengths For Public Keys Used For Exchanging Symmetric Keys*

RFC 4086, *Randomness Requirements for Security*

RFC 4106, *The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload*

RFC 4303, *IP Encapsulating Security Payload (ESP)*

RFC 4306, *Internet Key Exchange (IKEv2) Protocol*

RFC 4309, *Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload*

RFC 4434, *The AES-XCBC-PRF-128 Algorithm for the Internet Key Exchange Protocol (IKE)*

RFC 4595, *Use of IKEv2 in the Fibre Channel Security Association Management Protocol*

RFC 4718, *IKEv2 Clarifications and Implementation Guidelines*

RFC 4753, *ECP Groups for IKE and IKEv2*

RFC 4754, *IKE and IKEv2 Authentication Using the Elliptic Curve Digital Signature Algorithm (ECDSA)*

RFC 4868, *Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec*

## 2.6 OMG references

Copies of the following approved OMG specifications may be obtained through the Object Management Group (OMG) at [www.omg.org](http://www.omg.org).

*OMG Unified Modeling Language (UML) Specification Version 1.5, March 2003*



### 3 Definitions, symbols, abbreviations, and conventions

#### 3.1 Definitions

**3.1.1 access control list (ACL):** The data used by a SCSI target device to configure access rights for initiator ports according to the access controls state of the SCSI target device (see 8.3.1.3).

**3.1.2 access control list entry (ACE):** One entry in the access control list (see 3.1.1).

**3.1.3 access controls:** An optional SCSI target device feature that restricts initiator port access to specific logical units and modifies the information about logical units in the parameter data of the INQUIRY and REPORT LUNS commands (see 8.3.1).

**3.1.4 access controls coordinator:** The entity within a SCSI target device that coordinates the management and enforcement of access controls (see 8.3.1) for all logical units within the SCSI target device. The access controls coordinator is always addressable through the ACCESS CONTROLS well known logical unit (see 8.3) and LUN 0.

**3.1.5 active power condition:** When a device server is capable of responding to all of its supported commands, including media access requests, without delay. See 5.10.

**3.1.6 additional authenticated data (AAD):** Bytes of data input to a combined mode encryption algorithm (e.g., AES-GCM) as part of integrity checking computations (e.g., the IKEv2-SCSI Encrypted payload (see 7.6.3.5.10) uses AAD).

**3.1.7 additional sense code:** A combination of the ADDITIONAL SENSE CODE field and the ADDITIONAL SENSE CODE QUALIFIER field in sense data (see 4.5).

**3.1.8 alias list:** A list of alias values (see 3.1.9) and their associated designations (see 3.1.37) maintained by the device server and managed by the CHANGE ALIASES command (see 6.2) and REPORT ALIASES command (see 6.21).

**3.1.9 alias value:** A numeric value associated to a designation (see 3.1.37) in the alias list (see 3.1.8) and used in command or parameter data to reference a SCSI target device or SCSI target port. See 6.2.2.

**3.1.10 application client:** An object that is the source of SCSI commands. Further definition of an application client may be found in SAM-4.

**3.1.11 attribute:** A single unit of MAM (see 3.1.85) information.

**3.1.12 auto contingent allegiance (ACA):** The task set condition established following the return of CHECK CONDITION status when the NACA bit is set to one in the CONTROL byte. A detailed definition of ACA may be found in SAM-4.

**3.1.13 blocked task:** A task that is in the blocked state. Tasks become blocked when an ACA condition occurs. The blocked state ends when the ACA condition is cleared. A detailed definition of the blocked task state may be found in SAM-4.

**3.1.14 byte:** A sequence of eight contiguous bits considered as a unit.

**3.1.15 cache memory:** A temporary and often volatile data storage area outside the area accessible by application clients that may contain a subset of the data stored in the non-volatile data storage area.

**3.1.16 CbCS capability:** A descriptor that is a component of a CbCS credential (see 3.1.16) and a CbCS extension descriptor (see 5.14.6.8.16) that specifies access to a logical unit or volume (see SSC-3) for specific commands. See 5.14.6.8.13.

**3.1.17 CbCS capability key:** A shared key (see 3.1.155) that is cryptographically associated with a specific CbCS capability (see 3.1.16).

**3.1.18 CbCS credential:** The combination of a CbCS capability (see 3.1.16) and a CbCS capability key (see 3.1.17) that is returned by a RECEIVE CREDENTIAL command (see 6.19). See 5.14.6.8.12.

**3.1.19 CbCS Management Application Client class:** The CbCS (see 5.14.6.8) class whose objects manage, or an object that manages CbCS shared keys (see 5.14.6.8.11). See 5.14.6.8.4.

**3.1.20 CbCS Management Device Server class:** The CbCS (see 5.14.6.8) class whose objects prepare, or an object that prepares CbCS credentials (see 5.14.6.8.12) in response to RECEIVE CREDENTIAL command (see 6.19) requests from secure CDB originator (see 5.14.6.8.5). See 5.14.6.8.3.

**3.1.21 CbCS master key:** A shared key (see 3.1.155) from which CbCS working keys (see 3.1.24) are derived that is composed of an authentication key component and a generation key component. See 5.14.6.8.11.

**3.1.22 CbCS security token:** A random nonce (see 3.1.115) that is chosen by the enforcement manager (see 5.14.6.8.7) for a given I\_T nexus and returned to a secure CDB originator (see 5.14.6.8.5). See 5.14.6.8.10.

**3.1.23 CbCS working keys:** Shared keys (see 3.1.155) that are used to cryptographically associate a CbCS capability (see 3.1.16) with a CbCS capability key (see 3.1.17). See 5.14.6.8.11.

**3.1.24 code set enumeration:** A coded value used in one field to indicate the format of data in other fields or descriptors. See 4.4.3.

**3.1.25 command:** A request describing a unit of work to be performed by a device server. A detailed definition of a command may be found in SAM-4.

**3.1.26 command descriptor block (CDB):** The structure used to communicate commands from an application client to a device server (see 4.3). A CDB may have a fixed length of up to 16 bytes or a variable length of between 12 and 260 bytes.

**3.1.27 command standard:** A SCSI standard that defines the model, commands, and parameter data for a device type (e.g., SBC-2, SSC-2, SMC-2, MMC-4, or SES-2). See clause 1.

**3.1.28 company\_id:** Synonym for OUI (see 3.1.97).

**3.1.29 Control mode page:** A mode page that provides controls over SCSI features (e.g., task set management and error logging) that are applicable to all device types. See 7.4.6.

**3.1.30 Control Extension mode page:** A mode page that provides controls over SCSI features that are applicable to all device types. See 7.4.7.

**3.1.31 copy manager:** The device server that receives an EXTENDED COPY command and performs the operation requested.

**3.1.32 copy target device:** The name given by the EXTENDED COPY command (see 6.3) to a source or destination logical unit (i.e., a copy target device is a logical unit (see 3.1.79), not a SCSI target device (see 3.1.138)).

**3.1.33 cryptographic command sequence (CCS):** A defined sequence of SECURITY PROTOCOL IN commands (see 6.30) and SECURITY PROTOCOL OUT commands (see 6.31) that realize the cryptographic protocol of a specified security operation (e.g., the SECURITY PROTOCOL IN commands and SECURITY PROTOCOL OUT commands in the Key Exchange step and Authentication step, if any, of an IKEv2-SCSI (see 3.1.64) SA creation transaction (see 3.1.123)).

**3.1.34 data-in buffer:** The buffer specified by the application client to receive data from the device server during the processing of a command (see 4.2 and SAM-4).

**3.1.35 data-out buffer:** The buffer specified by the application client to supply data that is sent from the application client to the device server during the processing of a command (see 4.2 and SAM-4).

**3.1.36 deferred error:** A CHECK CONDITION status and sense data that is returned as the result of an error or exception condition that occurred during processing of a previous command for which GOOD status or CONDITION MET status has already been returned. See 4.5.5.

**3.1.37 designation:** When used in reference to access controls, a name and optional identifier information that specifies a SCSI target device or SCSI target port for association with an alias value (see 3.1.9) in the alias list (see 3.1.8). See 6.2.2. Otherwise, a distinguishing name, identifier, or title.

**3.1.38 Device Identification VPD page:** A VPD page that provides the means to retrieve identification information about the SCSI device, logical unit, and SCSI port. See 7.7.3.

**3.1.39 device server:** An object within a logical unit that processes SCSI tasks according to the rules of task management. A detailed definition of a device server may be found in SAM-4.

**3.1.40 device service request:** A request, submitted by an application client, conveying a SCSI command to a device server. A detailed definition of a device service request may be found in SAM-4.

**3.1.41 device service response:** The response returned to an application client by a device server on completion of a SCSI command. A detailed definition of a device service response may be found in SAM-4.

**3.1.42 device type:** The type of peripheral device (i.e., device model) implemented by the device server and indicated by the contents of the PERIPHERAL DEVICE TYPE field in the standard INQUIRY data (see 6.4.2).

**3.1.43 Disconnect-Reconnect mode page:** A mode page that provides the application client the means to tune the performance of a service delivery subsystem. See 7.4.8.

**3.1.44 element:** An addressable physical component of a medium changer SCSI device that may serve as the location of a removable unit of data storage medium. A detailed definition of an element may be found in SMC-2.

**3.1.45 enabled task state:** The only task state in which a task may make progress towards completion. A detailed definition of the enabled task state may be found in SAM-4.

**3.1.46 Encapsulating Security Payload for SCSI (ESP-SCSI):** A method for transferring encrypted and/or integrity checked parameter data in data-in buffers and/or data-out buffers based on Encapsulating Security Payload (see RFC 4303). See 5.14.7.

**3.1.47 Enforcement Manager class:** The command security (see 5.14.6) class whose objects are, or an object that is consulted by the Secure CDB Processor class (see 3.1.144) or Secure CDB Processor class objects to determine if the processing of a command is permitted or prohibited. Equivalent to the Policy Enforcement Point in the policy model defined by RFC 2753 and similar policy-based authorization standards. See 5.14.6.4.

**3.1.48 Error history I\_T nexus:** An I\_T nexus for which the device server has reserved access to the error history snapshot (see 3.1.49).

**3.1.49 Error history snapshot:** The contents of the error history at a specific point in time. See 5.12.2.

**3.1.50 Extended CDB (XCDB):** A CDB that contains another CDB and additional information to support extra processing (e.g., security features). See 4.3.4.

**3.1.51 Extended Unique Identifier, a 48-bit globally unique identifier (EUI-48):** The IEEE maintains a tutorial describing EUI-48 at <http://standards.ieee.org/regauth/oui/tutorials/EUI48.html>.

**3.1.52 Extended Unique Identifier, a 64-bit globally unique identifier (EUI-64):** The IEEE maintains a tutorial describing EUI-64 at <http://standards.ieee.org/regauth/oui/tutorials/EUI64.html>.

**3.1.53 faulted I\_T nexus:** The I\_T nexus on which a CHECK CONDITION status was returned that resulted in the establishment of an ACA. The faulted I\_T nexus condition is cleared when the ACA condition is cleared. See SAM-4.

**3.1.54 field:** A group of one or more contiguous bits, a part of a larger structure such as a CDB (see 3.1.26) or sense data (see 3.1.151).

**3.1.55 hard reset:** A condition resulting from the events defined by SAM-4 in which the SCSI device performs the hard reset operations described in SAM-4, this standard, and the applicable command standards.

**3.1.56 hashed message authentication code (HMAC):** A type of message authentication code that is calculated using the cryptographic hash function as defined in FIPS 198a (see 2.4) in combination with a secret key.

**3.1.57 host:** A SCSI device with the characteristics of a primary computing device, typically a personal computer, workstation, server, minicomputer, mainframe computer, or auxiliary computing device. A host includes one or more SCSI initiator devices.

**3.1.58 IEEE company\_id:** Synonym for OUI (see 3.1.97).

**3.1.59 I\_T nexus:** A nexus between a SCSI initiator port and a SCSI target port. See SAM-4.

**3.1.60 I\_T nexus loss:** A condition resulting from the events defined by SAM-4 in which the SCSI device performs the I\_T nexus loss operations described in SAM-4, this standard, and the applicable command standards.

**3.1.61 I\_T\_L nexus:** A nexus between a SCSI initiator port, a SCSI target port, and a logical unit. See SAM-4.

**3.1.62 I\_T\_L\_Q nexus transaction:** The information transferred between SCSI ports in a single data structure with defined boundaries (e.g., an information unit).

**3.1.63 idle power condition:** When a device server is capable of responding to all of its supported commands, including media access requests, but commands may take longer to complete than when in the active power condition. See 5.10.

**3.1.64 IKEv2-SCSI:** Internet Key Exchange protocol version 2 for SCSI. See 5.14.4.

**3.1.65 IKEv2-SCSI keys:** Shared keys (see 3.1.155) used to provide security for IKEv2-SCSI operations (e.g., creation and deletion of the SA). IKEv2-SCSI keys that are used after SA creation is complete are maintained in the MGMT\_DATA SA parameter (see 3.1.126). The shared keys names used by IKEv2-SCSI are listed in 5.14.4.4.

**3.1.66 IKEv2-SCSI CCS:** The CCS (see 3.1.33) that is the Key Exchange step and Authentication step, if any, of an IKEv2-SCSI (see 3.1.64) SA creation transaction (see 3.1.123).

**3.1.67 implicit head of queue:** An optional processing model for specified commands wherein the specified commands may be treated as if they had been received with a HEAD OF QUEUE task attribute (see SAM-4 and 5.3).

**3.1.68 information unit (IU):** A delimited and sequenced set of information in a format appropriate for transport by the service delivery subsystem (e.g., a CDB for a specific SCSI command).

**3.1.69 initiator port:** Synonymous with SCSI initiator port (see 3.1.133).

**3.1.70 initiator port identifier:** A value by which a SCSI initiator port (see 3.1.133) is referenced within a SCSI domain (see SAM-4).

**3.1.71 initiator port name:** A SCSI port name (see 3.1.136) of a SCSI initiator port (see 3.1.133) or of a SCSI target/initiator port when operating as a SCSI initiator port (see SAM-4).

**3.1.72 integrity check value:** A value used to cryptographically validate the integrity of a specified set of bytes that contain specified data based on a shared key (see 3.1.155).

**3.1.73 internet assigned numbers authority (IANA):** A service that assigns and manages registries of code values (i.e., code points) for the IETF (see <http://www.ietf.org>) and other users of the public internet. See <http://www.iana.org>. IANA maintains code values used by Internet Key Exchange version 2 (IKEv2) at <http://www.iana.org/assignments/ikev2-parameters>.

**3.1.74 Internet protocol domain name:** The name of a computer or hierarchy of computers within the domain name system defined by the IETF (see RFC 1035 and RFC 1591). The Internet Assigned Numbers Authority maintains a list of domain name assignments at <http://www.iana.org/assignments/domain-names>.

**3.1.75 Internet protocol number:** A coded value assigned to identify protocols that layer on the Internet protocol (see RFC 791). The Internet protocol number assigned to the transmission control protocol (TCP, see RFC 793) is six. The Internet Assigned Numbers Authority maintains a list of Internet protocol number assignments at <http://www.iana.org/assignments/protocol-numbers>.

**3.1.76 key derivation function (KDF):** An algorithm that is used to derive cryptographic keying material from a shared secret and other information.

**3.1.77 least significant bit (LSB):** In a binary code, the bit or bit position with the smallest numerical weighting in a group of bits that, when taken as a whole, represent a numerical value (e.g., in the number 0001b, the bit that is set to one).

**3.1.78 left-aligned:** A type of field containing ASCII data in which unused bytes are placed at the end of the field (highest offset) and are filled with ASCII space (20h) characters. See 4.4.1.

**3.1.79 logical unit:** An externally addressable entity within a SCSI target device that implements a SCSI device model and contains a device server. A detailed definition of a logical unit may be found in SAM-4.

**3.1.80 logical unit access control descriptor (LUACD):** The structure within an ACE (see 3.1.2) that identifies a logical unit to which access is allowed and specifies the LUN by which the logical unit is to be accessed (see 8.3.1.3.3).

**3.1.81 logical unit inventory:** The list of the logical unit numbers reported by a REPORT LUNS command (see 6.23).

**3.1.82 logical unit number (LUN):** An encoded 64-bit identifier for a logical unit. A detailed definition of a logical unit number may be found in SAM-4.

**3.1.83 logical unit reset:** A condition resulting from the events defined by SAM-4 in which the logical unit performs the logical unit reset operations described in SAM-4, this standard, and the applicable command standards.

**3.1.84 medium:** A physical entity that stores data in a nonvolatile manner (i.e., retained through a power cycle) in accordance with commands processed by the device server.

**3.1.85 medium auxiliary memory (MAM):** An auxiliary memory residing on a medium that is accessible to the device server (e.g., a tape cartridge). Medium auxiliary memory may be nonvolatile and independent of the main function of the device server.

**3.1.86 media changer:** A device that mechanizes the movement of media to and from the SCSI device that records on or reads from the media. A detailed definition of a medium changer may be found in SMC-2.

**3.1.87 most significant bit (MSB):** In a binary code, the bit or bit position with the largest numerical weighting in a group of bits that, when taken as a whole, represent a numerical value (e.g., in the number 1000b, the bit that is set to one).

**3.1.88 name:** A label of an object that is unique within a specified context and should never change (e.g., the term name and worldwide identifier (WWID) may be interchangeable).

**3.1.89 network address authority (NAA):** A field within a name (see 3.1.88) that specifies the format and length of that name. See 7.7.3.6 and FC-FS.

**3.1.90 nexus:** A relationship between two SCSI devices, and the SCSI initiator port and SCSI target port objects within those SCSI devices. See SAM-4.

**3.1.91 non-volatile cache memory:** Cache memory (see 3.1.15) that retains data through power cycles.

**3.1.92 nonce:** A value that is used one and only one time to provide uniqueness to a value (e.g., a secure cryptographic key) in whose derivation it participates or to uniquely identify a single instance of something (e.g., a timestamp) exchanged between an application client and a device server.

**3.1.93 null-padded:** A type of field in which unused bytes are placed at the end of the field (i.e., highest offset) and are filled with ASCII null (00h) characters. See 4.4.2.

**3.1.94 null-terminated:** A type of field in which the last used byte (i.e., highest offset) is required to contain an ASCII null (00h) character. See 4.4.2.

**3.1.95 Object Identifier (OID):** An ASN.1 format (see 2.2) identifier for an object in or related to a certificate. ISO/IEC 9834 describes procedures related to OID registration (see 2.2).

**3.1.96 one:** The logical true condition of a variable.

**3.1.97 organizationally unique identifier (OUI):** A numeric identifier that is assigned by the IEEE such that no assigned identifiers are identical. OUI is equivalent to company\_id or IEEE company\_id. The IEEE prefers OUI for EUI-48 identifiers (see 3.1.51) and company\_id for EUI-64 identifiers (see 3.1.52). However, the numeric identifier is called an OUI when it is assigned by the IEEE. The IEEE maintains a tutorial describing the OUI at <http://standards.ieee.org/regauth/oui/>.

**3.1.98 page:** A regular parameter structure (or format) used by several commands. These pages are identified with a value known as a page code.

**3.1.99 peripheral device:** An object that connects to a logical unit.

**3.1.100 peripheral device identifying information:** A value associated with a peripheral device accessed via the REPORT IDENTIFYING INFORMATION command (see 6.22) and SET IDENTIFYING INFORMATION command (see 6.33). See 5.15.

**3.1.101 peripheral device text identifying information:** A UTF-8 (see 3.1.180) format string associated with a peripheral device accessed via the REPORT IDENTIFYING INFORMATION command (see 6.22) and SET IDENTIFYING INFORMATION command (see 6.33). See 5.15.

**3.1.102 peripheral device type:** Synonym for device type (see 3.1.42).

**3.1.103 persist through power loss:** An optional capability associated with some features that allows an application client to request that a device server maintain information regarding that feature across power failures.

**3.1.104 persistent reservation holder:** The I\_T nexus(es) that are allowed to release or change a persistent reservation without preempting it. See 5.7.10.

**3.1.105 power cycle:** Power being removed from and later applied to a SCSI device.

**3.1.106 power on:** A condition resulting from the events defined by SAM-4 in which the SCSI device performs the power on operations described in SAM-4, this standard, and the applicable command standards.

**3.1.107 pre-shared key:** A shared key (see 3.1.155) that is established by a method outside the scope of this standard prior to the initiation of the function that requires its use. Requirement on pre-shared keys that are particular to this standard are described in 5.14.4.5. Some of the RFCs referenced by this standard use the name pre-shared secret for the same information that this standard calls a pre-shared key.

**3.1.108 primary target port asymmetric access state:** The characteristic that defines the behavior of a target port and the allowable command set for a logical unit when commands and task management functions are routed through the target port maintaining that state. See 5.9.2.1.

**3.1.109 primary target port group:** A set of target ports that are in the same primary target port asymmetric access state (see 3.1.108) at all times. See 5.9.2.1.

**3.1.110 primary target port group asymmetric access state:** The primary target port asymmetric access state (see 3.1.108) common to the set of target ports in a primary target port group (see 3.1.109). See 5.9.2.1.

**3.1.111 protocol identifier:** A coded value used in various fields to identify the protocol to which other fields apply. See 7.5.1.

**3.1.112 protocol specific:** A requirement that is defined by a SCSI transport protocol standard (see clause 1). A detailed definition of protocol specific may be found in SAM-4.

**3.1.113 protocol standard:** A SCSI standard that defines SCSI transport protocol (e.g., SAS, SPI-5, SBP-3, or FCP-2). See clause 1.

**3.1.114 proxy token:** An identifier for a logical unit that may be used to gain temporary access to that logical unit in the presence of access controls (see 8.3.1.6.2).

**3.1.115 random nonce:** A secure random number (see 4.6) that has a negligible chance of repeating and whose uses include providing significant uniqueness and randomness in cryptographic calculations.

**3.1.116 request for comment (RFC):** The name given to standards developed by the Internet Engineering Task Force (see 2.5).

**3.1.117 registered:** The condition that exists for an I\_T nexus following the successful completion of a PERSISTENT RESERVE OUT command with a REGISTER service action, REGISTER AND IGNORE EXISTING KEY service action (see 5.7.7), or REGISTER AND MOVE service action (see 5.7.8) and lasting until the registration is removed (see 5.7.11).

**3.1.118 registrant:** An I\_T nexus that is registered (see 3.1.117).

**3.1.119 right-aligned:** A type of field containing ASCII data in which unused bytes are placed at the start of the field (i.e., lowest offset) and are filled with ASCII space (20h) characters. See 4.4.1.

**3.1.120 relative port identifier:** An identifier for a SCSI port (see 3.1.134) that is unique within a SCSI device (see 3.1.129). See SAM-4. Application clients may use the SCSI Ports VPD page (see 7.7.7) to determine relative port identifier values.

**3.1.121 relative initiator port identifier:** A relative port identifier (see 3.1.120) for a SCSI initiator port (see 3.1.133).

**3.1.122 relative target port identifier:** A relative port identifier (see 3.1.120) for a SCSI target port (see 3.1.139).

**3.1.123 SA creation transaction:** Any sequence of SECURITY PROTOCOL IN commands (see 6.30) and SECURITY PROTOCOL OUT commands (see 6.31), including but not limited a CCS (see 3.1.33), that is used to create an SA between an application client and device server.

**3.1.124 SA generation:** Computation and initialization of the SA parameter values (see 3.1.126) required to create an SA (see 3.1.148) that is performed separately by the application client and device server after all SCSI commands required to create an SA have been performed without error. See 5.14.4.11.

**3.1.125 SA keys:** Shared keys (see 3.1.155) that are maintained in the USAGE\_DATA SA parameter (see 3.1.126) and used to provide security for the operations that use the SA (e.g., encryption of command parameter data). The shared keys names used by IKEv2-SCSI are listed in 5.14.4.4.

**3.1.126 SA parameters:** The parameters stored by both an application client and a device server that are associated with one SA (see 3.1.148) and identified by a pair of SAs (see 3.1.149). See 5.14.2.2.

**3.1.127 SA participant:** An application client or device server that participates in the creation or use of an SA (see 3.1.148).

**3.1.128 saturating counter:** A counter that remains at its maximum value after reaching its maximum value.

**3.1.129 SCSI device:** A device that contains one or more SCSI ports that are each connected to a service delivery subsystem and supports a SCSI application protocol (see SAM-4).

**3.1.130 SCSI device name:** A name (see 3.1.88) of a SCSI device that is world wide unique within the protocol of a SCSI domain (see 3.1.131) in which the SCSI device has SCSI ports (see 3.1.134). The SCSI device name may be made available to other SCSI devices or SCSI ports in protocol specific ways. See SAM-4.

**3.1.131 SCSI domain:** The interconnection of two or more SCSI devices and a service delivery subsystem. A detailed definition of a SCSI Domain may be found in SAM-4.



**3.1.132 SCSI initiator device:** A SCSI device (see 3.1.129) containing application clients (see 3.1.10) and SCSI initiator ports (see 3.1.133) that originate device service and task management requests (see SAM-4) to be processed by a SCSI target device (see 3.1.138) and receives device service and task management responses from SCSI target devices.

**3.1.133 SCSI initiator port:** A SCSI initiator device (see SAM-4) object that acts as the connection between application clients and a service delivery subsystem through which requests and responses are routed.

**3.1.134 SCSI port:** A port of a SCSI device that connects the application client, device server or task manager to a service delivery subsystem (see SAM-4).

**3.1.135 SCSI port identifier:** A value by which a SCSI port is referenced within a domain. The SCSI port identifier is either an initiator port identifier (see 3.1.170) or a target port identifier (see 3.1.170).

**3.1.136 SCSI port name:** A name (see 3.1.88) of a SCSI port that is world wide unique within the protocol of the SCSI domain of that SCSI port (see 3.1.134). The name may be made available to other SCSI devices or SCSI ports in that SCSI domain in protocol specific ways. See SAM-4.

**3.1.137 SCSI Ports VPD page:** A VPD page that allows retrieval of information about all the SCSI ports in a SCSI target device or SCSI target/initiator device. See 7.7.7.

**3.1.138 SCSI target device:** A SCSI device (see 3.1.129) containing logical units (see 3.1.79) and SCSI target ports (see 3.1.139) that receives device service and task management requests (see SAM-4) for processing and sends device service and task management responses to SCSI initiator devices.

**3.1.139 SCSI target port:** A SCSI target device (see SAM-4) object that acts as the connection between device servers and task managers and a service delivery subsystem through which requests and responses are routed.

**3.1.140 SCSI transport protocol standard:** A SCSI standard that defines a SCSI transport protocol (e.g., FCP-2, SAS, SRP, or SBP-3). See clause 1.

**3.1.141 secondary target port asymmetric access state:** A characteristic indicating a condition that affects the way in which an individual target port participates in its assigned primary target port group (see 3.1.109). See 5.9.2.1.

**3.1.142 secondary target port group:** One or more target ports that are in the same secondary target port asymmetric access state (see 3.1.141). See 5.9.2.1.

**3.1.143 Secure CDB Originator class:** The command security (see 5.14.6) class whose objects are, or an object that is an application client that originates secure CDBs and performs any additional functions necessary to do so. See 5.14.6.2.

**3.1.144 Secure CDB Processor class:** The command security (see 5.14.6) class whose objects are, or an object that is a device server that processes secure CDBs and performs any additional functions necessary to do so. See 5.14.6.3.

**3.1.145 Secure Digital Card Association (SD Card):** An organization that establishes and maintains standards for secure digital memory card applications. See <http://www.sdcard.org/>

**3.1.146 Secure hash algorithm (SHA):** A secure hash algorithm (e.g., SHA-1) specified in FIPS 180-2 with Change Notice 1 dated February 25, 2004 (see 2.4).

**3.1.147 Secure random number:** A random number that is generated in ways that protect it from security attacks. See 4.6.

**3.1.148 security association (SA):** A relationship and associated security processing between an application client and device server that is used to apply security functions (e.g., data integrity checking, data encryption) to data that is transferred in either direction. See 5.14.2.

**3.1.149 security association index (SAI):** A number representing the parameters for a security association as stored internally by the application client or device server. In other security models, this value is called the security parameters index (SPI). See 5.14.2.

**3.1.150 Security Manager class:** The command security (see 5.14.6) class whose objects maintain, or an object that maintains information about which secure CDBs may be originated by which secure CDB originator (see 3.1.143) application clients to which secure CDB processor (see 3.1.144) device servers, responds to requests to allow the origination of secure CDBs from secure CDB originator application clients, and updates secure CDB permissions and prohibitions in appropriate enforcement manager (see 3.1.47) members. Equivalent to the Policy Decision Point in the policy model defined by RFC 2753 and similar policy-based authorization standards. See 5.14.6.5.

**3.1.151 sense data:** Data describing an error or exceptional condition that a device server delivers to an application client in the same I\_T\_L\_Q nexus transaction (see 3.1.62) as a CHECK CONDITION status or as parameter data in response to a REQUEST SENSE command (see 6.29). The format of sense data is defined in 4.5.

**3.1.152 sense key:** The contents of the SENSE KEY field in sense data (see 4.5).

**3.1.153 service action:** A request describing a unit of work to be performed by a device server. A service action is an extension of a command. See SAM-4.

**3.1.154 service delivery subsystem:** That part of a SCSI domain that transmits service requests to a logical unit or SCSI target device and returns logical unit or SCSI target device responses to a SCSI initiator device. See SAM-4.

**3.1.155 shared key (SK):** A cryptographically protected secret (e.g., with more randomness (see RFC 4089) than a password) that is known only to a defined and limited set of entities (e.g., one application client and one device server). The shared keys names used by IKEv2-SCSI are listed in 5.14.4.4.

**3.1.156 SK\_ai:** Shared key (see 3.1.155) used for integrity checking data in a Data-Out Buffer (e.g., integrity checking the Encrypted payload in an IKEv2-SCSI Authentication step SECURITY PROTOCOL OUT command (see 5.14.4.9.2)).

**3.1.157 SK\_ar:** Shared key (see 3.1.155) used for integrity checking data in a Data-In Buffer (e.g., integrity checking the Encrypted payload in an IKEv2-SCSI Authentication step SECURITY PROTOCOL IN command (see 5.14.4.9.3)).

**3.1.158 SK\_d:** Shared key (see 3.1.155) KDF input material use to generate the shared keys stored in the KEYMAT SA Parameter (see 3.1.126).

**3.1.159 SK\_ei:** Shared key (see 3.1.155) used for encrypting data in a Data-Out Buffer (e.g., encrypting the Encrypted payload in an IKEv2-SCSI Authentication step SECURITY PROTOCOL OUT command (see 5.14.4.9.2)).

**3.1.160 SK\_er:** Shared key (see 3.1.155) used for encrypting data in a Data-In Buffer (e.g., encrypting the Encrypted payload in an IKEv2-SCSI Authentication step SECURITY PROTOCOL IN command (see 5.14.4.9.3)).

**3.1.161 SK\_pi:** Shared key (see 3.1.155) used to construct the IKEv2-SCSI Data-Out Buffer Authentication payload (see 5.14.4.9.2).

**3.1.162 SK\_pr:** Shared key (see 3.1.155) used to construct the IKEv2-SCSI Data-In Buffer Authentication payload (see 5.14.4.9.3).

**3.1.163 standby power condition:** When a device server is capable of accepting commands, but not capable of processing media access commands. See 5.10.

**3.1.164 status:** One byte of response information sent from a device server to an application client upon completion of each command. See SAM-4.

**3.1.165 storage networking industry association (SNIA):** An organization that develops and promotes standards for storage management and other storage-related functions, and which publishes standards via the INCITS Fast Track process. See <http://www.snia.org/>.

**3.1.166 system:** One or more SCSI domains operating as a single configuration.

**3.1.167 target port:** Synonymous with SCSI target port (see 3.1.139).

**3.1.168 target port asymmetric access state:** A primary target port asymmetric access state (see 3.1.108) or secondary target port asymmetric access state (see 3.1.141).

**3.1.169 target port group:** A primary target port group (see 3.1.109) or secondary target port group (see 3.1.142).

**3.1.170 target port identifier:** A value by which a SCSI target port (see 3.1.139) is referenced within a SCSI domain (see SAM-4).

**3.1.171 target port name:** A SCSI port name (see 3.1.136) of a SCSI target port or of a SCSI target/initiator port when operating as a SCSI target port (see SAM-4).

**3.1.172 task:** An object within a logical unit that represents the work associated with a command. A detailed definition of a task may be found in SAM-4.

**3.1.173 task set:** A group of tasks within a logical unit, whose interaction is dependent on the task management (queuing) and ACA rules. See SAM-4 and the Control mode page (see 7.4.6).

**3.1.174 TCP port numbers:** One of the data needed to establish a TCP connection. TCP port numbers may be assigned to protocols that layer on TCP by the Internet Assigned Numbers Authority. The Internet Assigned Numbers Authority maintains a list of TCP port number assignments at <http://www.iana.org/assignments/port-numbers>.

**3.1.175 third-party command:** A command sent to one SCSI device requesting that an operation be performed involving two other SCSI devices (e.g., the EXTENDED COPY command may perform copy operations between two or more SCSI devices none of which are the SCSI device to which the EXTENDED COPY command was sent).

**3.1.176 Trusted Computing Group (TCG):** An organization that develops and promotes open standards for hardware-enabled trusted computing and security technologies. See <https://www.trustedcomputinggroup.org>.

**3.1.177 unit attention condition:** Asynchronous status information that a logical unit establishes to report to the initiator ports associated with one or more I\_T nexuses. See SAM-4.

**3.1.178 universal time (UT):** The time at longitude zero, colloquially known as Greenwich Mean Time. See <http://aa.usno.navy.mil/faq/docs/UT.html>.

**3.1.179 URI Schemes:** The Internet Assigned Numbers Authority maintains a list of schemes for URI and URL names at <http://www.iana.org/assignments/uri-schemes>.

**3.1.180 UTF-8:** A character set that is a transformation format of the character set defined by ISO 10646. See RFC 2279.

**3.1.181 vendor specific (VS):** Something (e.g., a bit, field, or code value) that is not defined by this standard and may be vendor defined.

**3.1.182 volatile cache memory:** Cache memory (see 3.1.15) that does not retain data through power cycles.

**3.1.183 well known logical unit:** A logical unit that only does specific functions (see clause 8). Well known logical units allow an application client to issue requests to receive and manage specific information usually relating to a SCSI target device.

**3.1.184 well known logical unit number (W-LUN):** The logical unit number that identifies a well known logical unit.

**3.1.185 wrapping counter:** A counter that wraps back to zero after reaching its maximum value.

**3.1.186 zero:** The logical false condition of a variable.

**3.1.187 zero-padded:** A type of field in which unused bytes are placed at the end of the field (i.e., highest offset) and are filled with zeros. See 4.4.2.

## 3.2 Symbols and acronyms

÷	divided by
×	multiplied by
<	less than
>	greater than
	concatenation
x, xx	any valid value for a bit or field
AAD	Additional Authenticated Data (see 3.1.6)
ACE	Access Control list Entry (see 3.1.2)
ACL	Access Control List (see 3.1.1)
ACA	Auto Contingent Allegiance (see 3.1.12)
ADC	Automation/Drive Interface - Commands (see clause 1)
ADC-2	Automation/Drive Interface - Commands -2 (see clause 1)
ADC-3	Automation/Drive Interface - Commands -3 (see clause 2)
ADT	Automation/Drive Interface - Transport Protocol (see clause 1)
ADT-2	Automation/Drive Interface - Transport Protocol -2 (see clause 1)
AES	Advanced Encryption Standards
AES-GCM	Advanced Encryption Standard - Galois Counter Mode (see RFC 4106)
ASC	Additional Sense Code (see 4.5)
ASCII	American Standard Code for Information Interchange (see 2.2)
ASCQ	Additional Sense Code Qualifier (see 4.5)
ASN.1	Abstract Syntax Notation One (see 2.2)
ATA	AT Attachment (see <a href="http://www.t13.org">www.t13.org</a> )
ATAPI	AT Attachment with Packet Interface (see <a href="http://www.t13.org">www.t13.org</a> )
CbCS	Capability-based Command Security (see 5.14.6.8)
CCS	Cryptographic Command Sequence (see 3.1.33)

CDB	Command Descriptor Block (see 3.1.26)
CRC	Cyclic Redundancy Check
D_ID	Destination Identifier (defined in FC-FS, see clause 1)
ECP	Elliptic Curve group modulo Prime (see RFC 4753)
ECDSA	Elliptic Curve Digital Signature Algorithm (see RFC 4754)
ESP-SCSI	Encapsulating Security Payload for SCSI (see 3.1.46)
EUI-48	Extended Unique Identifier, a 48-bit globally unique identifier (see 3.1.51)
EUI-64	Extended Unique Identifier, a 64-bit globally unique identifier (see 3.1.52)
FC-FS	Fibre Channel Framing and Signaling Interface (see clause 1)
FCP-2	Fibre Channel Protocol for SCSI -2 (see clause 1)
FCP-3	Fibre Channel Protocol for SCSI -3 (see clause 1)
HMAC	Hashed Message Authentication Code (see 3.1.56)
HTTP	Hypertext Transfer Protocol (see RFC 2616)
IANA	Internet Assigned Numbers Authority (see 3.1.73)
ID	Identifier or Identification
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force (see 2.5)
IKEv2	Internet Key Exchange version 2 (see 2.5)
IP	Internet Protocol (see 2.5)
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
iSCSI	Internet SCSI (see 2.5)
ISO	Organization for International Standards
IU	Information Unit
KDF	Key Derivation Function (see 3.1.76)
LBA	Logical Block Address
LSB	Least Significant Bit (see 3.1.77)
LUACD	Logical Unit Access Control Descriptor (see 3.1.80)
LUN	Logical Unit Number (see 3.1.82)
MAM	Medium Auxiliary Memory (see 3.1.85)
MMC-4	SCSI Multi-Media Commands -4 (see clause 1)
MMC-5	SCSI Multi-Media Commands -5 (see clause 1)
MMC-6	SCSI Multi-Media Commands -6 (see clause 1)
MODP	Modular exponential (see RFC 3526)
MSB	Most Significant Bit (see 3.1.87)
NAA	Network Address Authority (see 3.1.89)
n/a	not applicable
INCITS	InterNational Committee for Information Technology Standards
OCRW	SCSI Specification for Optical Card Reader/Writer (see clause 1)
OID	Object Identifier (see 3.1.95)
OSD	Object-based Storage Devices Commands (see clause 1)
OUI	Organizationally Unique Identifier (see 3.1.97)
PKI	Public Key Infrastructure (see RFC 3280)
PRF	Pseudo-Random Function (see RFC 4306)
RAID	Redundant Array of Independent Disks
RBC	SCSI Reduced Block Commands (see clause 1)
RDMA	Remote Direct Memory Access (see SRP)
RFC	Request For Comments (see 3.1.116)
RMC	SCSI Reduced Multi-Media Commands (see clause 1)
SA	Security Association (see 3.1.148)
SAI	Security Association Index (see 3.1.149)
SAM-2	SCSI Architecture Model -2 (see clause 1)
SAM-3	SCSI Architecture Model -3 (see clause 1)

SAM-4	SCSI Architecture Model -4 (see clause 1)
SAT	SCSI / ATA Translation (see clause 1)
SAT-2	SCSI / ATA Translation -2 (see clause 1)
SAUT	SA Usage Type (see 7.6.3.5.13)
SBC-2	SCSI Block Commands -2 (see clause 1)
SBC-3	SCSI Block Commands -3 (see clause 1)
SBP-3	Serial Bus Protocol -3 (see clause 1)
SCC-2	SCSI Controller Commands -2 (see clause 1)
SCSI	The architecture defined by the family of standards described in clause 1
SES	SCSI-3 Enclosure Services (see clause 1)
SES-2	SCSI Enclosure Services -2 (see clause 1)
SHA-1	Secure Hash Algorithm, 160 bits (see 3.1.146)
SHA-256	Secure Hash Algorithm, 256 bits (see 3.1.146)
SHA-384	Secure Hash Algorithm, 384 bits (see 3.1.146)
SHA-512	Secure Hash Algorithm, 512 bits (see 3.1.146)
SK	Shared Key (see 3.1.155)
SMC-2	SCSI Media Changer Commands -2 (see clause 1)
SMC-3	SCSI Media Changer Commands -3 (see clause 1)
SNIA	Storage Networking Industry Association (see 3.1.165)
SPC	SCSI-3 Primary Commands (ANSI INCITS 301-1997, see clause 1)
SPC-2	SCSI Primary Commands -2 (see clause 1)
SPC-3	SCSI Primary Commands -2 (this standard, see clause 1)
SPI-5	SCSI Parallel Interface -5 (see clause 1)
SRP	SCSI RDMA Protocol (see clause 1)
SSC-2	SCSI Stream Commands -2 (see clause 1)
SSC-3	SCSI Stream Commands -3 (see clause 1)
TCG	Trusted Computing Group (see 3.1.176)
TCP	Transmission Control Protocol (see RFC 793)
UML	Unified Modeling Language (see 2.6)
URI	Uniform Resource Identifier (see RFC 2396, RFC 3305, and 3.1.179)
URL	Uniform Resource Locator (see RFC 2396, RFC 3305, and 3.1.179)
UT	Universal time (see 3.1.178)
USB	Universal Serial Bus (see <a href="http://www.usb.org">www.usb.org</a> )
VPD	Vital Product Data (see 7.7)
VS	Vendor Specific (see 3.1.181)
W-LUN	Well known logical unit number (see 3.1.184)
XCDB	Extended CDB (see 3.1.50)

### 3.3 Keywords

**3.3.1 expected:** A keyword used to describe the behavior of the hardware or software in the design models assumed by this standard. Other hardware and software design models may also be implemented.

**3.3.2 ignored:** A keyword used to describe an unused bit, byte, word, field or code value. The contents or value of an ignored bit, byte, word, field or code value shall not be examined by the receiving SCSI device and may be set to any value by the transmitting SCSI device.

**3.3.3 invalid:** A keyword used to describe an illegal or unsupported bit, byte, word, field or code value. Receipt of an invalid bit, byte, word, field or code value shall be reported as an error.

**3.3.4 mandatory:** A keyword indicating an item that is required to be implemented as defined in this standard.

**3.3.5 may:** A keyword that indicates flexibility of choice with no implied preference (equivalent to "may or may not").

**3.3.6 may not:** A keyword that indicates flexibility of choice with no implied preference (equivalent to "may or may not").

**3.3.7 obsolete:** A keyword indicating that an item was defined in prior SCSI standards but has been removed from this standard.

**3.3.8 optional:** A keyword that describes features that are not required to be implemented by this standard. However, if any optional feature defined by this standard is implemented, then it shall be implemented as defined in this standard.

**3.3.9 prohibited:** A keyword used to describe a feature, function, or coded value that is defined in a non-SCSI standard (i.e., a standard that is not a member of the SCSI family of standards) to which this standard makes a normative reference where the use of said feature, function, or coded value is not allowed for implementations of this standard.

**3.3.10 reserved:** A keyword referring to bits, bytes, words, fields and code values that are set aside for future standardization. A reserved bit, byte, word or field shall be set to zero, or in accordance with a future extension to this standard. Recipients are not required to check reserved bits, bytes, words or fields for zero values. Receipt of reserved code values in defined fields shall be reported as an error.

**3.3.11 restricted:** A keyword referring to bits, bytes, words, and fields that are set aside for use in other SCSI standards. A restricted bit, byte, word, or field shall be treated as a reserved bit, byte, word or field for the purposes of the requirements defined in this standard.

**3.3.12 shall:** A keyword indicating a mandatory requirement. Designers are required to implement all such mandatory requirements to ensure interoperability with other products that conform to this standard.

**3.3.13 should:** A keyword indicating flexibility of choice with a strongly preferred alternative; equivalent to the phrase "it is strongly recommended".

**3.3.14 x or xx:** The value of the bit or field is not relevant.

## 3.4 Conventions

Certain words and terms used in this standard have a specific meaning beyond the normal English meaning. These words and terms are defined either in 3.1 or in the text where they first appear. Names of commands, statuses, sense keys, and additional sense codes are in all uppercase (e.g., REQUEST SENSE). Lowercase is used for words having the normal English meaning.

If there is more than one CDB length for a particular command (e.g., MODE SENSE(6) and MODE SENSE(10)) and the name of the command is used in a sentence without any CDB length descriptor (e.g., MODE SENSE), then the condition specified in the sentence applies to all CDB lengths for that command.

The names of fields are in small uppercase (e.g., ALLOCATION LENGTH). When a field name is a concatenation of acronyms, uppercase letters may be used for readability (e.g., NORMACA). Normal case is used when the contents of a field are being discussed. Fields containing only one bit are usually referred to as the name bit instead of the name field.

A binary number is represented in this standard by any sequence of digits consisting of only the Western-Arabic numerals 0 and 1 immediately followed by a lower-case b (e.g., 0101b). Underscores or spaces may be included in binary number representations to increase readability or delineate field boundaries (e.g., 0 0101 1010b or 0\_0101\_1010b).

A hexadecimal number is represented in this standard by any sequence of digits consisting of only the Western-Arabic numerals 0 through 9 and/or the upper-case English letters A through F immediately followed by a lower-case h (e.g., FA23h). Underscores or spaces may be included in hexadecimal number representations to increase readability or delineate field boundaries (e.g., B FD8C FA23h or B\_FD8C\_FA23h).

A decimal number is represented in this standard by any sequence of digits consisting of only the Western-Arabic numerals 0 through 9 not immediately followed by a lower-case b or lower-case h (e.g., 25).

A range of numeric values is represented in this standard in the form "a to z", where a is the first value included in the range, all values between a and z are included in the range, and z is the last value included in the range (e.g., the representation "0h to 3h" includes the values 0h, 1h, 2h, and 3h).

When the value of the bit or field is not relevant, x or xx appears in place of a specific value.

This standard uses the following conventions for representing decimal numbers:

- a) The decimal separator (i.e., separating the integer and fractional portions of the number) is a period;
- b) The thousands separator (i.e., separating groups of three digits in a portion of the number) is a space; and
- c) The thousands separator is used in both the integer portion and the fraction portion of a number.

Table 1 shows some examples of decimal numbers represented using various conventions.

**Table 1 — Numbering conventions examples**

French	English	This standard
0,6	0.6	0.6
3,141 592 65	3.14159265	3.141 592 65
1 000	1,000	1 000
1 323 462,95	1,323,462.95	1 323 462.95

A decimal number represented in this standard with an overline over one or more digits following the decimal point is a number where the overlined digits are infinitely repeating (e.g.,  $666.\overline{6}$  means 666.666 666... or  $666 \frac{2}{3}$  and  $12.\overline{142\ 857}$  means 12.142 857 142 857... or  $12 \frac{1}{7}$ ).

Lists sequenced by lowercase or uppercase letters show no ordering relationship between the listed items.

EXAMPLE 1 - The following list shows no relationship between the colors named:

- a) red, specificity one of the following colors:
  - a) crimson; or
  - b) amber;
- b) blue; or
- c) green.

Lists sequenced by numbers show an ordering relationship between the listed items.

EXAMPLE 2 - The following list shows the order in which a page is meant to be read:

- 1) top;



- 2) middle; and
- 3) bottom.

If a conflict arises between text, tables, or figures, the order of precedence to resolve the conflicts is text; then tables; and finally figures. Not all tables or figures are fully described in the text. Tables show data format and values. Notes do not constitute any requirements for implementors.

### 3.5 Bit and byte ordering

This subclause describes the representation of fields in a table that defines the format of a SCSI structure (e.g., the format of a CDB).

If a field consists of more than one bit and contains a single value (e.g., a number), the least significant bit (LSB) is shown on the right and the most significant bit (MSB) is shown on the left (e.g., in a byte, bit 7 is the MSB and is shown on the left, and bit 0 is the LSB and is shown on the right). The MSB and LSB are not labeled if the field consists of 8 or fewer bits.

If a field consists of more than one byte and contains a single value, the byte containing the MSB is stored at the lowest address and the byte containing the LSB is stored at the highest address (i.e., big-endian byte ordering). The MSB and LSB are labeled.

If a field consists of more than one byte and contains multiple fields each with their own values (e.g., a descriptor), there is no MSB and LSB of the field itself and thus there are no MSB and LSB labels. Each individual field has an MSB and LSB that are labeled as appropriate in the table, if any, that describes the format of the sub-structure having multiple fields.

If a field contains a text string (e.g., ASCII or UTF-8), the MSB label is the MSB of the first character and the LSB label is the LSB of the last character.

When required for clarity, multiple byte fields may be represented with only two rows in a table. This condition is represented by values in the byte number column not increasing by one in each subsequent table row, thus indicating the presence of additional bytes.

### 3.6 Notation conventions

#### 3.6.1 Notation for byte encoded character strings

When this standard requires one or more bytes to contain specific encoded characters, the specific characters are enclosed in double quotation marks. The double quotation marks identify the start and end of the characters that are required to be encoded but are not themselves to be encoded. The characters that are to be encoded are shown in the case that is to be encoded.

The encoded characters and the double quotation marks that enclose them are preceded by text that specifies the character encoding methodology and the number of characters required to be encoded.

Using the notation described in this subclause, stating that eleven ASCII characters “SCSI device” are to be encoded would be the same writing out the following sequence of byte values: 53h 43h 53h 49h 20h 64h 65h 76h 69h 63h 65h.

### 3.6.2 Notation for procedure calls

In this standard, the model for functional interfaces between objects is a procedure call. Such interfaces are specified using the following notation:

[Result =] Procedure Name (IN ([input-1] [,input-2] ...), OUT ([output-1] [,output-2] ...))

Where:

Result: A single value representing the outcome of the procedure call.

Procedure Name: A descriptive name for the function modeled by the procedure call. When the procedure call model is used to describe a SCSI transport protocol service, the procedure name is the same as the service name.

Input-1, Input-2, ...: A comma-separated list of names identifying caller-supplied input arguments.

Output-1, Output-2, ...: A comma-separated list of names identifying output arguments to be returned by the procedure call.

"[ ...]": Brackets enclosing optional or conditional arguments.

This notation allows arguments to be specified as inputs and outputs. The following is an example of a procedure call specification:

Found = Search (IN (Pattern, Item List), OUT ([Item Found]))

Where:

Found = Flag

Flag: if set to one, indicates that a matching item was located.

Input Arguments:

Pattern = ... /\* Definition of Pattern argument \*/  
Argument containing the search pattern.

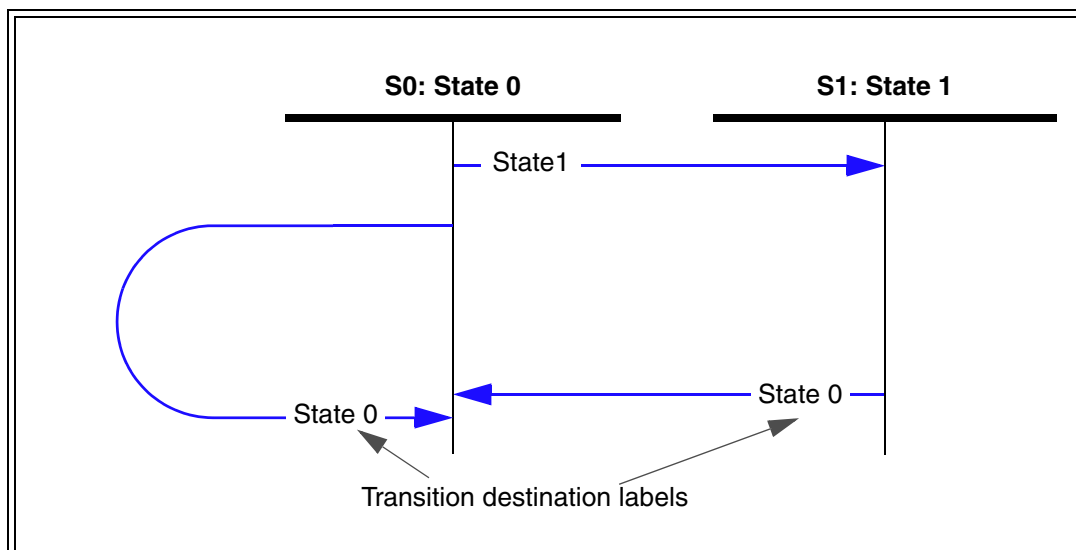
Item List = Item<NN> /\* Definition of Item List as an array of NN Item arguments\*/  
Contains the items to be searched for a match.

Output Arguments:

Item Found = Item ... /\* Item located by the search procedure call \*/  
This argument is only returned if the search succeeds.

### 3.6.3 Notation for state diagrams

All state diagrams use the notation shown in figure 2.



**Figure 2 — Example state diagram**

The state diagram is followed by subclauses describing the states and state transitions.

Each state and state transition is described in the list with particular attention to the conditions that cause the transition to occur and special conditions related to the transition.

A system specified in this manner has the following properties:

- a) Time elapses only within discrete states; and
- b) State transitions are logically instantaneous.

### 3.6.4 Notation for binary power multipliers

The nomenclature used for binary power multiplier values in this standard (see table 2) is based on IEC 60027:2000, Letter symbols to be used in electrical technology - Part 2: Telecommunications and electronics.

**Table 2 — Binary power multiplier nomenclature**

Prefix	Abbreviation	Power of two	Example
kibi	Ki	$2^{10}$ or 1 024	one kibibit is 1 Kib is 1 024 bits
mebi	Mi	$2^{20}$	one mebibyte is 1 MiB is 1 048 576 bytes
gebi	Gi	$2^{30}$	one gibibyte is 1 GiB is 1 073 741 824 bytes
tebi	Ti	$2^{40}$	
pebi	Pi	$2^{50}$	
exbi	Ei	$2^{60}$	

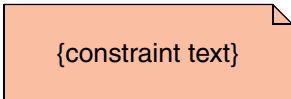
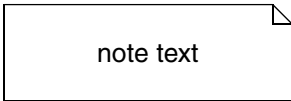
### 3.6.5 Notation for UML figures

#### 3.6.5.1 Introduction

This standard contains class diagram figures that use notation that is based on UML (see 2.6).

Some class diagrams contain constraints or notes that use the notion shown in table 3.

**Table 3 — Class diagram constraints and notes notation**

Notation	Description
	The presence of the curly brackets defines constraint that is a normative requirement. An example of a constraint is shown in figure 3.
	The absence of curly brackets defines a note that is informative. An example of a note is shown in figure 5.

The notation used to denote multiplicity in class diagrams is shown in table 4.

**Table 4 — Class diagram multiplicity notation**

Notation	Description
not specified	The number of instances of an attribute is not specified.
1	One instance of the class or attribute exists.
0..*	Zero or more instances of the class or attribute exist.
1..*	One or more instances of the class or attribute exist.
0..1	Zero or one instance of the class or attribute exists.
n..m	n to m instances of the class or attribute exist (e.g., 2..8).
x,n..m	Multiple disjoint instances of the class or attribute exist (e.g., 2, 8..15).

Class diagrams show:

- a) Two or more classes (see 3.6.5.2); and
- b) One or more of the following relationships between them:
  - a) Association (see 3.6.5.3);
  - b) Aggregation (see 3.6.5.4);
  - c) Generalization (see 3.6.5.5); and
  - d) Dependency (see 3.6.5.6).

### 3.6.5.2 Class notation

The notation used for classes is shown in table 5.

**Table 5 — Class diagram notation for classes**

Notation			Description
<b>Class Name</b>	<b>Class Name</b>	<b>Class Name</b>	A class with no attributes or operations
	<b>Class Name</b>	<b>Class Name</b>	A class with attributes and no operations
	Attribute01[1]	Attribute01[1]	
	Attribute02[1]	Attribute02[1]	
		<b>Class Name</b>	A class with operations and no attributes
		Operation01() Operation02()	
		<b>Class Name</b>	A class with attributes and operations
		Attribute01[1]	
		Attribute02[1]	
		Operation01() Operation02()	
		<b>Class Name</b>	A class with attributes that have a specified multiplicity (see table 4 in 3.6.5.1) and operations
		Attribute01[1..*]	
		Attribute02[1]	
		Operation01() Operation02()	

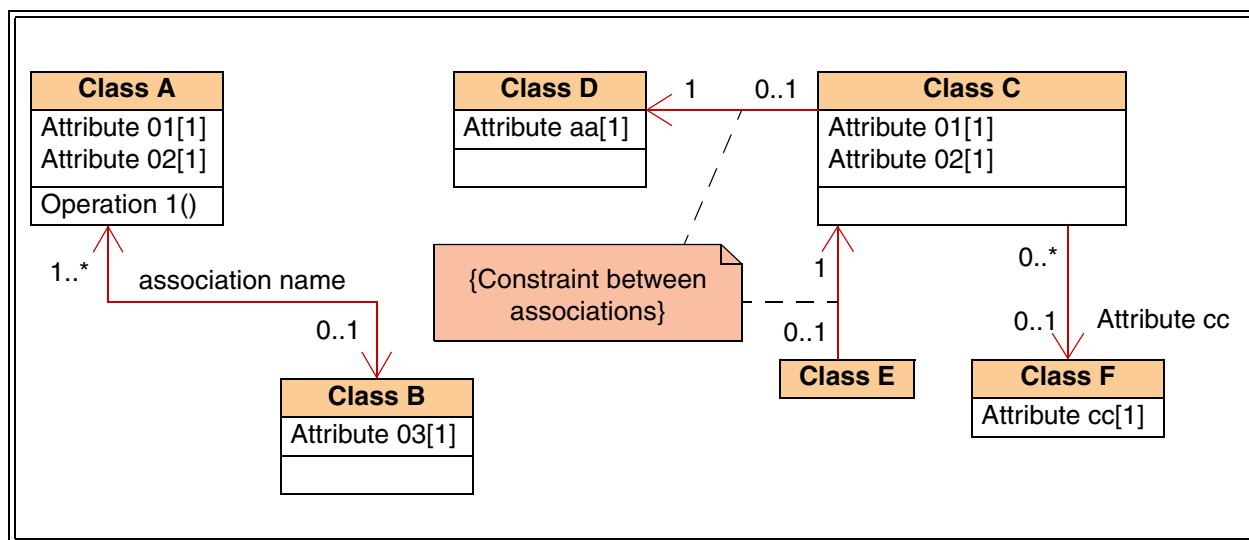
### 3.6.5.3 Class association relationships notation

The notation used to denote association (i.e., “knows about”) relationships between classes is shown in table 6. Unless the two classes in an association relationship also have an aggregation relationship (see 3.6.5.4), association relationships have multiplicity notation (see table 4 in 3.6.5.1) at each end of the relationship line.

**Table 6 — Class diagram notation for associations**

Notation	Description
	Class A knows about Class B (i.e., read as “Class A association name Class B”) and Class B knows about Class A (i.e., read as “Class B association name Class A”)
	Class B knows about Class A (i.e., read as “Class B knows about Class A”) but Class A does not know about Class B
	Class A knows about Class B (i.e., read as “Class A uses the role name attribute of Class B”) but Class B does not know about Class A
Note: The use of role names and association names are optional.	

Several example association relationships between classes are shown in figure 3.

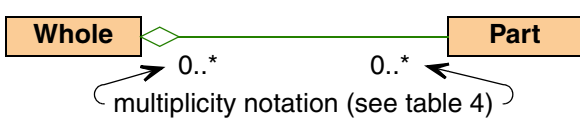



**Figure 3 — Example class association relationships**

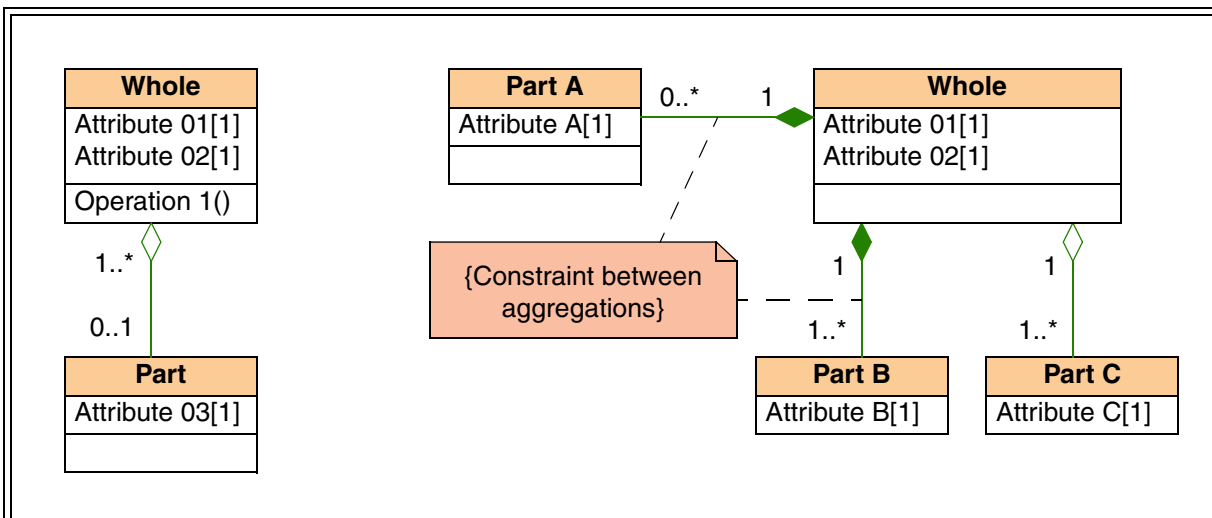
### 3.6.5.4 Class aggregation relationships notation

The aggregation relationship is a specific type of association. The notation used to denote aggregation (i.e., “is a part of” or “contains”) relationships between classes is shown in table 7. Aggregation relationships always include multiplicity notation (see table 4 in 3.6.5.1) at each end of the relationship line.

**Table 7 — Class diagram notation for aggregations**

Notation	Description
	The Part class is part of the Whole class and may continue to exist even if the Whole class is removed (i.e., read as “the Whole contains the Part.”)
	The Part class is part of the Whole class, shall only belong to one Whole class, and shall not continue to exist if the Whole class is removed (i.e., read as “the Whole contains the Part.”)

Several example aggregation relationships between classes are shown in figure 4.




**Figure 4 — Example class aggregation relationships**

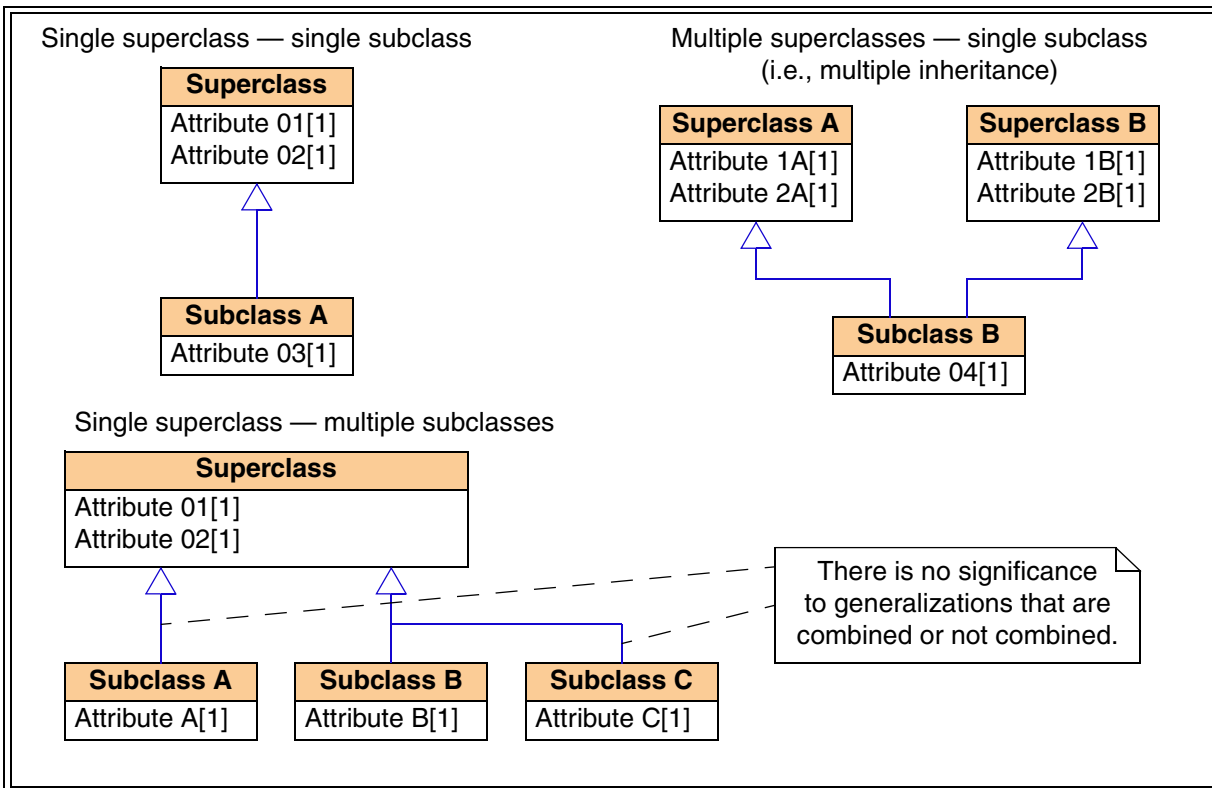
### 3.6.5.5 Class generalization relationships notation

The notation used to denote generalization (i.e., “is a kind of”) relationships between classes is shown in table 8.

**Table 8 — Class diagram notation for generalizations**

Notation	Description
	Subclass is a kind of superclass. A subclass shares all the attributes and operations of the superclass (i.e., the subclass inherits from the superclass).

Several example generalization relationships between classes are shown in figure 5.




**Figure 5 — Example class generalization relationships**



### 3.6.5.6 Class dependency relationships notation

The notation used to denote dependency (i.e., “depends on”) relationships between classes is shown in table 9.

**Table 9 — Class diagram notation for dependencies**

Notation	Description
	Class A depends on class B. A change in class B may cause a change in class A.

An example dependency relationship between classes is shown in figure 6.

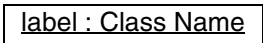
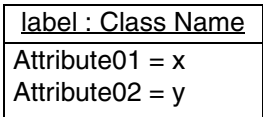
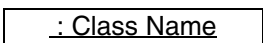
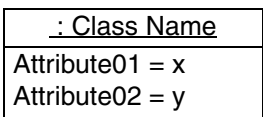


**Figure 6 — Example class dependency relationships**

### 3.6.5.7 Object notation

The notation used for objects is shown in table 10.

**Table 10 — Notation for objects**

Notation	Description
	Notation for a named object with no attributes
	Notation for a named object with attributes
	Notation for a anonymous object with no attributes
	Notation for a anonymous object with attributes

## 4 General Concepts

### 4.1 Introduction

This standard defines behaviors that are common to all SCSI device models (see clause 5). This standard defines the SCSI commands that are basic to more than one device model and the SCSI commands that may apply to any device model (see clause 6). This standard defines the parameters that are basic to more than one device model (see clause 7).

### 4.2 The request-response model

The SCSI command set assumes an underlying request-response protocol. The fundamental properties of the request-response protocol are defined in SAM-4. Action on SCSI commands shall not be deemed completed until a response is received. The response shall include a status that indicates the final disposition of the command. As per SAM-4, the request-response protocol may be modeled as a procedure call, specifically:

Service response = Execute Command (IN (I\_T\_L\_Q Nexus, CDB, Task Attribute, [Data-In Buffer Size], [Data-Out Buffer], [Data-Out Buffer Size], [Command Reference Number], [Command Priority]), OUT ([Data-In Buffer], [Sense Data], [Sense Data Length], Status))

SAM-4 defines all of the inputs and outputs in the procedure call above. As they may apply to any SCSI device, this standard defines the contents of the following procedure inputs and outputs; CDB, Data-Out Buffer, Data-In Buffer, and Sense Data. This standard does not define all possible instances of these procedure inputs and outputs. This standard defines only those instances that may apply to any SCSI device. Instances of the procedure inputs and outputs that apply to specific SCSI device models are defined in the applicable SCSI command standards (see 3.1.27).

This standard references values returned via the Status output parameter (e.g., CHECK CONDITION, RESERVATION CONFLICT). Status values are not defined by this standard. SAM-4 defines all Status values.

The entity that makes the procedure call is an application client (see SAM-4). The procedure call's representation arrives at the SCSI target device in the form of a device service request. The entity that performs the work of the procedure call is a device server (see SAM-4).

### 4.3 The Command Descriptor Block (CDB)

#### 4.3.1 CDB usage and structure

A command is communicated by sending a command descriptor block (CDB) to the device server. For several commands, the CDB is accompanied by a list of parameters in the Data-Out Buffer. See the specific commands for detailed information.

If a logical unit validates reserved CDB fields and receives a reserved field within the CDB that is not zero, then the logical unit shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

If a logical unit receives a reserved CDB code value in a field other than the OPERATION CODE field, then the logical unit shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The fixed length CDB formats are described in 4.3.2. The variable length CDB formats are described in 4.3.3. The XCDB format is described in 4.3.4. The CDB fields that are common to most commands are described in 4.3.5. The fields shown in 4.3.2 and 4.3.3 and described in 4.3.5 are used consistently by most commands. However, the actual usage of any field (except the OPERATION CODE field and the CONTROL field) is described in the subclause defining that command. If a device server receives a CDB containing an operation code that is invalid or not supported, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID COMMAND OPERATION CODE.

For all commands, if there is an invalid parameter in the CDB, the device server shall terminate the command without altering the medium.

The XCDB format (see 4.3.4) is a CDB in which additional information is appended to another CDB. The requirements in this subclause apply to the XCDB and the CDB contained in the XCDB.

#### 4.3.2 The fixed length CDB formats

All fixed length CDBs shall have an OPERATION CODE field as their first byte and a CONTROL byte as their last byte. Table 11 shows the typical format of a 6-byte CDB. Table 12 shows the typical format of a 10-byte CDB. Table 13 shows the typical format of a 12-byte CDB. Table 14 shows the typical format of a 16-byte CDB. Table 15 shows the format of a 16-byte CDB for commands that provide for a long LBA.

**Table 11 — Typical CDB for 6-byte commands**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	miscellaneous CDB information			(MSB)				
2	LOGICAL BLOCK ADDRESS (if required)							
3								
4	TRANSFER LENGTH (if required) PARAMETER LIST LENGTH (if required) ALLOCATION LENGTH (if required)							
5	CONTROL							

**Table 12 — Typical CDB for 10-byte commands**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	miscellaneous CDB information			SERVICE ACTION (if required)				
2	(MSB)							
3								
4	LOGICAL BLOCK ADDRESS (if required)							
5	(LSB)							
6	miscellaneous CDB information							
7	(MSB)							
8	TRANSFER LENGTH (if required) PARAMETER LIST LENGTH (if required) ALLOCATION LENGTH (if required)							
9	(LSB)							
	CONTROL							

**Table 13 — Typical CDB for 12-byte commands**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	miscellaneous CDB information			SERVICE ACTION (if required)				
2	(MSB)							
3	LOGICAL BLOCK ADDRESS (if required)							
4								
5								
5								
6	(MSB)							
7	TRANSFER LENGTH (if required)							
8	PARAMETER LIST LENGTH (if required)							
9	ALLOCATION LENGTH (if required)							
9	(LSB)							
10	miscellaneous CDB information							
11	CONTROL							

Table 14 — Typical CDB for 16-byte commands

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	miscellaneous CDB information			SERVICE ACTION (if required)				
2	(MSB)							
3								
4	LOGICAL BLOCK ADDRESS (if required)							
5								
6								
7								
8	Additional CDB data (if required)							
9								
10	(MSB)							
11	TRANSFER LENGTH (if required)							
12	PARAMETER LIST LENGTH (if required)							
13	ALLOCATION LENGTH (if required)							
14	miscellaneous CDB information							
15	CONTROL							

Table 15 — Typical CDB for long LBA 16-byte commands

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	miscellaneous CDB information							
2	(MSB)	LOGICAL BLOCK ADDRESS						
3								
4								
5								
6								
7								
8								
9								(LSB)
10	(MSB)	TRANSFER LENGTH (if required) PARAMETER LIST LENGTH (if required) ALLOCATION LENGTH (if required)						
11								
12								
13								(LSB)
14	miscellaneous CDB information							
15	CONTROL							

### 4.3.3 The variable length CDB formats

The first byte of a variable length CDB shall contain the operation code 7Fh. The CONTROL byte is the second byte in the variable length CDB (see table 16).

**Table 16 — Typical variable length CDB**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2	miscellaneous CDB information							
3	miscellaneous CDB information							
4	miscellaneous CDB information							
5	miscellaneous CDB information							
6	miscellaneous CDB information							
7	ADDITIONAL CDB LENGTH (n-7)							
8	(MSB)							
9	SERVICE ACTION							
10	(LSB)							
n	Service action specific fields							

The ADDITIONAL CDB LENGTH field specifies the number of additional CDB bytes. This value in the ADDITIONAL CDB LENGTH field shall be a multiple of 4. If the number of CDB bytes delivered by the service delivery subsystem is not sufficient to contain the number of bytes specified by the ADDITIONAL CDB LENGTH field, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The SERVICE ACTION field specifies the action being requested by the application client. The SERVICE ACTION field is required in the variable length CDB format and is described in 4.3.5.2. Each service action code description defines a number of service action specific fields that are needed for that service action.

A 32-byte variable length CDB format (see table 17) is defined for long LBA operations.

**Table 17 — Typical variable length CDB for long LBA 32-byte commands**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2	miscellaneous CDB information							
3	miscellaneous CDB information							
4	miscellaneous CDB information							
5	miscellaneous CDB information							
6	miscellaneous CDB information							
7	ADDITIONAL CDB LENGTH (18h)							
8	(MSB)	SERVICE ACTION						
9								(LSB)
10	miscellaneous CDB information			DPO	FUA	miscellaneous CDB information		
11	miscellaneous CDB information							
12	(MSB)	LOGICAL BLOCK ADDRESS						
19								(LSB)
20	miscellaneous CDB information							
27								
28	(MSB)	TRANSFER LENGTH (if required) PARAMETER LIST LENGTH (if required) ALLOCATION LENGTH (if required)						
31								(LSB)

#### 4.3.4 Extended CDBs

##### 4.3.4.1 XCDB model

With one exception, any SCSI CDB may be extended in an XCDB. An XCDB shall not be extended in another XCDB. Instead, additional XCDB descriptors shall be added to the existing XCDB.

XCDB descriptors may be:

- Added by application clients and removed by device servers, or
- Added and removed by entities outside the scope of this standard that transport or process CDBs and XCDBs during their transfer from an application client to a device server.



#### 4.3.4.2 The XCDB format

The first byte of an XCDB shall contain the operation code 7Eh. The CONTROL byte is the CONTROL byte in the CDB field (see table 18).

**Table 18 — XCDB format**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Eh)							
1	Reserved							
2	(MSB) _____							
3	LENGTH (n-3) _____ (LSB)							
4	_____							
i	CDB _____							
	XCDB descriptors							
k	_____							
	XCDB descriptor [first] _____							
	⋮							
	_____							
n	XCDB descriptor [last] _____							

The LENGTH field contains the number of bytes that follow in the XCDB.

The CDB field contains the CDB to which XCDB descriptors are being appended.

Each XCDB descriptor (see table 19) contains fields associated with a specified extension type (see table 20). The command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID XCDB, if:

- a) An XCDB does not contain at least one XCDB descriptor;
- b) More than one XCDB descriptor contains the same extension type; or
- c) The order of extension types in the XCDB descriptors differs from that shown in table 20.

**Table 19 — XCDB descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	EXTENSION TYPE							
1	Extension parameters							
n								

The EXTENSION TYPE field (see table 20) specifies the size and format of the extension parameters that follow in the XCDB descriptor.

**Table 20 — EXTENSION TYPE field**

Code	Descriptor Order <sup>a</sup>	Description	Extension size (bytes)	Reference
40h	first	CbCS extension descriptor	140	5.14.6.8.16
all others	Reserved			
<sup>a</sup> The order in which XCDB descriptors appear in an XCDB is arranged so that all the XCDB descriptors that follow an XCDB descriptor defined in a future version of this standard are also XCDB descriptors defined in a future version of this standard (i.e., after encountering one unrecognized XCDB descriptor, all subsequent XCDB descriptors are also going to be unrecognized).				

The extension parameters contain additional information whose processing is associated with the CDB in the CDB field. The number, content, and size of the extension parameters depends on the contents of the EXTENSION TYPE field.

#### 4.3.5 Common CDB fields

##### 4.3.5.1 Operation code

The first byte of a SCSI CDB shall contain an operation code identifying the operation being requested by the CDB. Some operation codes provide for modification of their operation based on a service action (see 4.3.5.2). In such cases, the operation code and service action code combine to identify the operation being requested. The location of the SERVICE ACTION field in the CDB varies depending on the operation code value.

The OPERATION CODE (see table 21) of the CDB has a GROUP CODE field and a COMMAND CODE field. The three-bit GROUP CODE field provides for eight groups of command codes. The five-bit COMMAND CODE field provides for thirty-two command codes in each group. A total of 256 possible operation codes exist. Operation codes are defined in this standard and other command standards (see 3.1.27). The group code value (see table 22) shall determine the length of the CDB.

**Table 21 — OPERATION CODE byte**

Bit	7	6	5	4	3	2	1	0
	GROUP CODE			COMMAND CODE				

The value in the GROUP CODE field specifies one of the groups shown in table 22.

**Table 22 — Group Code values**

Group Code	Meaning	Typical CDB format
000b	6 byte commands	see table 11 in 4.3.2
001b	10 byte commands	see table 12 in 4.3.2
010b	10 byte commands	see table 12 in 4.3.2
011b	reserved <sup>a</sup>	
100b	16 byte commands	see table 14 and table 15 in 4.3.2
101b	12 byte commands	see table 13 in 4.3.2
110b	vendor specific	
111b	vendor specific	
<sup>a</sup> The format of the commands using the group code 011b and operation code 7Fh is described in 4.3.3. The format of the commands using the group code 011b and operation code 7Eh is described in 4.3.4. With the exception of operation codes 7Fh and 7Eh, all group code 011b operation codes are reserved.		

#### 4.3.5.2 Service action

All CDB formats except the 6-byte format provide for a SERVICE ACTION field containing a coded value identifying a function to be performed under the more general command function specified in the OPERATION CODE field. While the SERVICE ACTION field is defined for CDB formats, it is used as described in this subclause only in those CDB formats that contain a SERVICE ACTION field. When the specific field SERVICE ACTION is not defined in a CDB format, the bits identified as the SERVICE ACTION field in a CDB shall be used or reserved as specified by the particular CDB format.

#### 4.3.5.3 Logical block address

The logical block addresses on a logical unit or within a volume or partition shall begin with block zero and be contiguous up to the last logical block of that logical unit or within that volume or partition.

A six-byte CDB may contain a 21-bit LOGICAL BLOCK ADDRESS field. The ten-byte and the twelve-byte CDBs may contain 32-bit LOGICAL BLOCK ADDRESS fields. The sixteen-byte CDB has two formats: one allows a 32-bit LOGICAL BLOCK ADDRESS field (see table 14) and the other allows a 64-bit LOGICAL BLOCK ADDRESS field (see table 15). LOGICAL BLOCK ADDRESS fields in additional parameter data have their length specified for each occurrence. See the specific command descriptions.

#### 4.3.5.4 Transfer length

The TRANSFER LENGTH field specifies the amount of data to be transferred, usually the number of blocks. Some commands use transfer length to specify the requested number of bytes to be sent as defined in the command description.

Commands that use one byte for the TRANSFER LENGTH field may allow up to 256 blocks or 256 bytes of data to be transferred by one command.

In commands that use multiple bytes for the TRANSFER LENGTH field, a transfer length of zero specifies that no data transfer shall take place. A value of one or greater specifies the number of blocks or bytes that shall be transferred.

Refer to the specific command description for further information.

#### 4.3.5.5 Parameter list length

The PARAMETER LIST LENGTH field is used to specify the number of bytes sent from the Data-Out Buffer. This field is typically used in CDBs for parameters that are sent to a device server (e.g., mode parameters, diagnostic parameters, log parameters). A parameter length of zero specifies that no data shall be transferred. This condition shall not be considered as an error, unless otherwise specified.

#### 4.3.5.6 Allocation length

The ALLOCATION LENGTH field specifies the maximum number of bytes or blocks that an application client has allocated in the Data-In Buffer. The ALLOCATION LENGTH field specifies bytes unless a different requirement is stated in the command definition.

An allocation length of zero specifies that no data shall be transferred. This condition shall not be considered as an error.

The device server shall terminate transfers to the Data-In Buffer when the number of bytes or blocks specified by the ALLOCATION LENGTH field have been transferred or when all available data have been transferred, whichever is less. The allocation length is used to limit the maximum amount of variable length data (e.g., mode data, log data, diagnostic data) returned to an application client. If the information being transferred to the Data-In Buffer includes fields containing counts of the number of bytes in some or all of the data, then the contents of these fields shall not be altered to reflect the truncation, if any, that results from an insufficient ALLOCATION LENGTH value, unless the standard that describes the Data-In Buffer format states otherwise.

If the amount of information to be transferred exceeds the maximum value that the ALLOCATION LENGTH field is capable of specifying, the device server shall transfer no data and terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

#### 4.3.5.7 Control

The contents of the CONTROL byte are defined in SAM-4. The CONTROL byte has the same definition for all commands.

### 4.4 Data field requirements

#### 4.4.1 ASCII data field requirements

ASCII data fields shall contain only ASCII printable characters (i.e., code values 20h to 7Eh) and may be terminated with one or more ASCII null (00h) characters.

ASCII data fields described as being left-aligned shall have any unused bytes at the end of the field (i.e., highest offset) and the unused bytes shall be filled with ASCII space characters (20h).

ASCII data fields described as being right-aligned shall have any unused bytes at the start of the field (i.e., lowest offset) and the unused bytes shall be filled with ASCII space characters (20h).

#### 4.4.2 Null data field termination and zero padding requirements

A data field that is described as being null-terminated shall have one byte containing an ASCII or UTF-8 null (00h) character in the last used byte (i.e., highest offset) of the field and no other bytes in the field shall contain the ASCII/UTF-8 null character.

A data field may be specified to be a fixed length. The length specified for a data field may be greater than the length required to contain the contents of the field. A data field may be specified to have a length that is a multiple of a given value (e.g., a multiple of four bytes). When such fields are described as being null-padded, the bytes at the end of the field that are not needed to contain the field data shall contain ASCII or UTF-8 null (00h) characters. When such fields are described as being zero-padded, the bytes at the end of the field that are not needed to contain the field data shall contain zeros.

NOTE 1 - There is no difference between the pad byte contents in null-padded and zero-padded fields. The difference is in the format of the other bytes in the field.

A data field that is described as being both null-terminated and null-padded shall have at least one byte containing an ASCII or UTF-8 null (00h) character in the end of the field (i.e., highest offset) and may have more than one byte containing ASCII or UTF-8 null characters, if needed, to meet the specified field length requirements. If more than one byte in a null-terminated, null-padded field contains the ASCII or UTF-8 null character, then all the bytes containing the ASCII or UTF-8 null character shall be at the end of the field (i.e., only the highest offsets).

#### 4.4.3 Variable type data field requirements

Parameter lists may contain fields or descriptors in which data may be represented in different formats. To indicate which format is being used a field may be defined that contains a code set enumeration (see table 23).

**Table 23 — Code set enumeration**

Code	Description
0h	Reserved
1h	The associated fields or descriptors contain binary values
2h	The associated fields or descriptors contain ASCII printable characters (i.e., code values 20h to 7Eh)
3h	The associated fields or descriptors contain UTF-8 (see 3.1.180) codes
4h to Fh	Reserved

### 4.5 Sense data

#### 4.5.1 Sense data introduction

Sense data shall be returned in the same I\_T\_L\_Q nexus transaction (see 3.1.62) as a CHECK CONDITION status and as parameter data in response to the REQUEST SENSE command (see 6.29). Sense data returned in the same I\_T\_L\_Q nexus transaction as a CHECK CONDITION status shall be either fixed or descriptor format sense data format based on the value of the D\_SENSE bit in the Control mode page (see 7.4.6). The REQUEST SENSE command may be used to request either the fixed format sense data or the descriptor format sense data.

The first byte of all sense data contains the RESPONSE CODE field (see table 24) that indicates the error type and format of the sense data.

**Table 24 — Sense data response codes**

Response Code	Error type		Sense data format	
	Description	Reference	Description	Reference
00h to 6Fh	Reserved			
70h	Current	4.5.4	Fixed	4.5.3
71h	Deferred	4.5.5	Fixed	4.5.3
72h	Current	4.5.4	Descriptor	4.5.2
73h	Deferred	4.5.5	Descriptor	4.5.2
74h to 7Eh	Reserved			
7Fh	Vendor specific			

The RESPONSE CODE field shall be set to 70h in all unit attention condition sense data in which:

- a) The ADDITIONAL SENSE CODE field is set to 29h; or
- b) The additional sense code is set to MODE PARAMETERS CHANGED.

## 4.5.2 Descriptor format sense data

### 4.5.2.1 Descriptor format sense data overview

The descriptor format sense data for response codes 72h (current errors) and 73h (deferred errors) is defined in table 25.

**Table 25 — Descriptor format sense data**

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved	RESPONSE CODE (72h or 73h)						
1	Reserved				SENSE KEY			
2	ADDITIONAL SENSE CODE							
3	ADDITIONAL SENSE CODE QUALIFIER							
4	Reserved							
6								
7	ADDITIONAL SENSE LENGTH (n-7)							
	Sense data descriptor(s)							
8	Sense data descriptor 0 (see table 26)							
	⋮							
	⋮							
n	Sense data descriptor x (see table 26)							

The contents of the RESPONSE CODE field indicate the error type and format of the sense data (see 4.5.1). For descriptor format sense data, the RESPONSE CODE field shall be set to 72h or 73h.

The SENSE KEY, ADDITIONAL SENSE CODE and ADDITIONAL SENSE CODE QUALIFIER fields provide a hierarchy of information. The hierarchy provides a top-down approach for an application client to determine information relating to the error and exception conditions.

The SENSE KEY field indicates generic information describing an error or exception condition. The sense keys are defined in 4.5.6.

The ADDITIONAL SENSE CODE (ASC) field indicates further information related to the error or exception condition reported in the SENSE KEY field. Support of the additional sense codes not required by this standard is optional. A list of additional sense codes is in 4.5.6. If the device server does not have further information related to the error or exception condition, the additional sense code shall be set to zero.

The ADDITIONAL SENSE CODE QUALIFIER (ASCQ) field indicates detailed information related to the additional sense code. If the error or exception condition is reported by the device server, the value returned shall be as specified in 4.5.6. If the device server does not have detailed information related to the error or exception condition, the additional sense code qualifier shall be set to zero.

The ADDITIONAL SENSE LENGTH field indicates the number of additional sense bytes that follow. The additional sense length shall be less than or equal to 244 (i.e., limiting the total length of the sense data to 252 bytes). If the sense

data is being returned as parameter data by a REQUEST SENSE command, then the relationship between the ADDITIONAL SENSE LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

Sense data descriptors (see table 26) provide specific sense information. A given type of sense data descriptor shall be included in the sense data only when the information it contains is valid.

**Table 26 — Sense data descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE							
1	ADDITIONAL LENGTH (n-1)							
2	Sense data descriptor specific							
n								

The DESCRIPTOR TYPE field (see table 27) identifies the type of sense data descriptor. No more than one sense data descriptor of each type shall be included in the descriptor format sense data.

**Table 27 — DESCRIPTOR TYPE field**

Code	Description	Reference
00h	Information	4.5.2.2
01h	Command specific information	4.5.2.3
02h	Sense key specific	4.5.2.4
03h	Field replaceable unit	4.5.2.5
04h	Stream commands	SSC-3
05h	Block commands	SBC-3
06h	OSD object identification	OSD
07h	OSD response integrity check value	OSD
08h	OSD attribute identification	OSD
09h	ATA Status Return	SAT
0Ah to 7Fh	Reserved	
80h to FFh	Vendor specific	4.5.2.6

The ADDITIONAL LENGTH field indicates the number of sense data descriptor specific bytes that follow in the sense data descriptor.



#### 4.5.2.2 Information sense data descriptor

The information sense data descriptor (see table 28) provides information that is device-type or command specific and is defined in a command standard (see 3.1.27).

**Table 28 — Information sense data descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE (00h)							
1	ADDITIONAL LENGTH (0Ah)							
2	VALID (1b)	Reserved						
3	Reserved							
4	INFORMATION							
11								

The DESCRIPTOR TYPE and ADDITIONAL LENGTH fields are described in 4.5.2.1. For the information sense data descriptor, the DESCRIPTOR TYPE field shall be set to 00h and the ADDITIONAL LENGTH field shall be set to 0Ah.

The VALID bit shall be set to one.

NOTE 2 - In previous versions of this standard and in the fixed format sense data, the VALID bit indicates whether the contents of the INFORMATION field is valid as defined by a command standard. Since the contents of the INFORMATION field are valid whenever an information sense data descriptor is included in the sense data, the only legal value for the VALID bit is set to one.

The contents of the INFORMATION field are device-type or command specific and are defined in a command standard (see 3.1.27). When a four byte quantity is stored in the INFORMATION field, the first four bytes shall be zero.

#### 4.5.2.3 Command-specific information sense data descriptor

The command-specific information sense data descriptor (see table 29) provides information that depends on the command on which the exception condition occurred.

**Table 29 — Command-specific information sense data descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE (01h)							
1	ADDITIONAL LENGTH (0Ah)							
2	Reserved							
3	Reserved							
4	COMMAND-SPECIFIC INFORMATION							
11								

The DESCRIPTOR TYPE and ADDITIONAL LENGTH fields are described in 4.5.2.1. For the command-specific information sense data descriptor, the DESCRIPTOR TYPE field shall be set to 01h and the ADDITIONAL LENGTH field shall be set to 0Ah.

The COMMAND-SPECIFIC INFORMATION field contains information that depends on the command on which the exception condition occurred. When a four byte quantity is stored in the COMMAND-SPECIFIC INFORMATION field, the first four bytes shall be zero.

Further meaning for the COMMAND-SPECIFIC INFORMATION field is defined within the command description in the appropriate command standard (e.g., see SBC-3 for the REASSIGN BLOCKS commands, or see 6.3 for the EXTENDED COPY command).

#### 4.5.2.4 Sense key specific sense data descriptor

##### 4.5.2.4.1 Sense key specific sense data descriptor introduction

The sense key specific sense data descriptor (see table 30) provides additional information about the exception condition. The format and content of the sense-key specific data depends on the value in the SENSE KEY field (see 4.5.2.1).

**Table 30 — Sense key specific sense data descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE (02h)							
1	ADDITIONAL LENGTH (06h)							
2	Reserved							
3	Reserved							
4	SKSV (1b)							
5								
6								
7	Reserved							

The DESCRIPTOR TYPE and ADDITIONAL LENGTH fields are described in 4.5.2.1. For the sense-key specific sense data descriptor, the DESCRIPTOR TYPE field shall be set to 01h and the ADDITIONAL LENGTH field shall be set to 06h.

The sense-key specific valid (SKSV) bit shall be set to one.

NOTE 3 - In previous versions of this standard and in the fixed format sense data, the sksv bit indicates whether the contents of the SENSE KEY SPECIFIC field are valid as defined by a command standard. Since the contents of the SENSE KEY SPECIFIC field are valid whenever a sense key specific sense data descriptor is included in the sense data, the only legal value for the SKSV bit is set to one.

The definition of the SENSE KEY SPECIFIC field (see table 31) is determined by the value of the SENSE KEY field (see 4.5.2.1).

**Table 31 — Sense key specific sense data descriptor definitions**

Sense Key	SENSE KEY SPECIFIC Field Definition	Reference
ILLEGAL REQUEST	Field pointer	4.5.2.4.2
HARDWARE ERROR, MEDIUM ERROR, or RECOVERED ERROR	Actual retry count	4.5.2.4.3
NO SENSE or NOT READY	Progress indication	4.5.2.4.4
COPY ABORTED	Segment pointer	4.5.2.4.5
UNIT ATTENTION	Unit attention condition queue overflow	4.5.2.4.6
All other sense keys	The sense key specific sense data descriptor shall not appear in the descriptor format sense data and the SKSV bit (see 4.5.3) shall be set to zero in the fixed format sense data.	

#### 4.5.2.4.2 Field pointer sense key specific data

If the sense key is ILLEGAL REQUEST, then the SENSE KEY SPECIFIC field shall be as shown in table 32.

**Table 32 — Field pointer sense key specific data**

Bit Byte	7	6	5	4	3	2	1	0
0	SKSV (1b)	C/D	Reserved		BPV	BIT POINTER		
1	(MSB)							
2	FIELD POINTER							(LSB)

The SKSV bit is described in 4.5.2.4.1 for descriptor format sense data and in 4.5.3 for fixed format sense data.

A command data (C/D) bit set to one indicates that the illegal parameter is in the CDB. A C/D bit set to zero indicates that the illegal parameter is in the data parameters sent by the application client in the Data-Out Buffer.

A bit pointer valid (BPV) bit set to zero indicates that the value in the BIT POINTER field is not valid. A BPV bit set to one indicates that the BIT POINTER field specifies which bit of the byte designated by the FIELD POINTER field is in error. When a multiple-bit field is in error, the BIT POINTER field shall point to the first bit (i.e., the left-most bit) of the field.

The FIELD POINTER field indicates which byte of the CDB or of the parameter data was in error. Bytes are numbered starting from zero, as shown in the tables describing the commands and parameters. When a multiple-byte field is in error, the field pointer shall point to the first byte (i.e., the left-most byte) of the field. If several consecutive bytes are reserved, each shall be treated as a single-byte field.

NOTE 4 - The bytes identified as being in error are not necessarily the bytes that need to be changed to correct the problem.

#### 4.5.2.4.3 Actual retry count sense key specific data

If the sense key is **HARDWARE ERROR**, **MEDIUM ERROR**, or **RECOVERED ERROR**, then the **SENSE KEY SPECIFIC** field shall be as shown in table 33.

**Table 33 — Actual retry count sense key specific data**

Bit Byte	7	6	5	4	3	2	1	0
0	sksv (1b)	Reserved						
1	(MSB)							
2		ACTUAL RETRY COUNT (LSB)						

The SKSV bit is described in 4.5.2.4.1 for descriptor format sense data and in 4.5.3 for fixed format sense data.

The ACTUAL RETRY COUNT field returns vendor specific information on the number of retries of the recovery algorithm used in attempting to recover an error or exception condition.

NOTE 5 - This field should be computed in the same way as the retry count fields within the Read-Write Error Recovery mode page (see SBC-2, SSC-3, and MMC-4).

#### 4.5.2.4.4 Progress indication sense key specific data

If the sense key is **NO SENSE** or **NOT READY**, the **SENSE KEY SPECIFIC** field shall be as shown in table 34.

**Table 34 — Progress indication sense key specific data**

Bit Byte	7	6	5	4	3	2	1	0
0	sksv (1b)	Reserved						
1	(MSB)	PROGRESS INDICATION						
2								
								(LSB)

The SKSV bit is described in 4.5.2.4.1 for descriptor format sense data and in 4.5.3 for fixed format sense data.

The PROGRESS INDICATION field is a percent complete indication in which the returned value is a numerator that has 65 536 (10000h) as its denominator. The progress indication shall be based upon the total operation.

NOTE 6 - The progress indication should be time related, however this is not an absolute requirement. (E.g., since format time varies with the number of defects encountered, etc., it is reasonable for the device server to assign values to various steps within the process. The granularity of these steps should be small enough to provide reasonable assurances to the application client that progress is being made.)

#### 4.5.2.4.5 Segment pointer sense key specific data

If the sense key is COPY ABORTED, the SENSE KEY SPECIFIC field shall be as shown in table 35.

**Table 35 — Segment pointer sense key specific data**

Bit Byte	7	6	5	4	3	2	1	0
0	SKSV (1b)	Reserved	SD	Reserved	BPV	BIT POINTER		
1	(MSB)	FIELD POINTER						
2								(LSB)

The SKSV bit is described in 4.5.2.4.1 for descriptor format sense data and in 4.5.3 for fixed format sense data.

The segment descriptor (SD) bit indicates whether the field pointer is relative to the start of the parameter list or to the start of a segment descriptor. An SD bit set to zero indicates that the field pointer is relative to the start of the parameter list. An SD bit set to one indicates that the field pointer is relative to the start of the segment descriptor indicated by the third and fourth bytes of the COMMAND-SPECIFIC INFORMATION field (see 6.3.3).

A bit pointer valid (BPV) bit set to zero indicates that the value in the BIT POINTER field is not valid. A BPV bit set to one indicates that the BIT POINTER field specifies which bit of the byte designated by the FIELD POINTER field is in error. When a multiple-bit field is in error, the BIT POINTER field shall point to the most-significant (i.e., left-most) bit of the field.

The FIELD POINTER field indicates which byte of the parameter list or segment descriptor was in error.

If the parameter list is in excess of 65 528 bytes in length and SD is set to zero, the FIELD POINTER value may not fit in two bytes provided by the sense key specific sense data descriptor.

#### 4.5.2.4.6 Unit attention condition queue overflow sense key specific data

If the sense key is UNIT ATTENTION, the SENSE KEY SPECIFIC field shall be as shown in table 36.

**Table 36 — Unit attention condition queue overflow sense key specific data**

Bit Byte	7	6	5	4	3	2	1	0
0	SKSV (1b)	Reserved						OVERFLOW
1		Reserved						
2								

The SKSV bit is described in 4.5.2.4.1 for descriptor format sense data and in 4.5.3 for fixed format sense data.

An OVERFLOW bit set to one indicates that the unit attention condition queue has overflowed. An OVERFLOW bit set to zero indicates that the unit attention condition queue has not overflowed.

#### 4.5.2.5 Field replaceable unit sense data descriptor

The field replaceable unit sense data descriptor (see table 37) provides information about a component that has failed.

**Table 37 — Field replaceable unit sense data descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE (03h)							
1	ADDITIONAL LENGTH (02h)							
2	Reserved							
3	FIELD REPLACEABLE UNIT CODE							

The DESCRIPTOR TYPE and ADDITIONAL LENGTH fields are described in 4.5.2.1. For the field replaceable unit sense data descriptor, the DESCRIPTOR TYPE field shall be set to 03h and the ADDITIONAL LENGTH field shall be set to 02h.

Non-zero values in the FIELD REPLACEABLE UNIT CODE field are used to identify a component that has failed. A value of zero in this field indicates that no specific component has been identified to have failed or that the data is not available. The format of this information is not specified by this standard. Additional information about the field replaceable unit may be available in the ASCII Information VPD page (see 7.7.2), if supported by the device server.

#### 4.5.2.6 Vendor specific sense data descriptors

Vendor specific sense data descriptors (see table 38) contain vendor specific data that further defines the nature of the exception condition.

**Table 38 — Vendor specific sense data descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE (80h to FFh)							
1	ADDITIONAL LENGTH (n-1)							
2	Vendor specific							
n								

The DESCRIPTOR TYPE and ADDITIONAL LENGTH fields are described in 4.5.2.1. For the vendor specific sense data descriptor, the DESCRIPTOR TYPE field shall be set to a value between 80h and FFh, inclusive.

### 4.5.3 Fixed format sense data

The fixed format sense data for response codes 70h (current errors) and 71h (deferred errors) is defined in table 39.

**Table 39 — Fixed format sense data**

Bit Byte	7	6	5	4	3	2	1	0
0	VALID	RESPONSE CODE (70h or 71h)						
1	Obsolete							
2	FILEMARK	EOM	ILI	Reserved	SENSE KEY			
3	INFORMATION							
6								
7	ADDITIONAL SENSE LENGTH (n-7)							
8	COMMAND-SPECIFIC INFORMATION							
11								
12	ADDITIONAL SENSE CODE							
13	ADDITIONAL SENSE CODE QUALIFIER							
14	FIELD REPLACEABLE UNIT CODE							
15	SKSV	SENSE KEY SPECIFIC						
17								
18	Additional sense bytes							
n								

A VALID bit set to zero indicates that the INFORMATION field is not defined in this standard or any other command standard (see 3.1.27). A VALID bit set to one indicates the INFORMATION field contains valid information as defined in this standard or a command standard.

The contents of the RESPONSE CODE field indicate the error type and format of the sense data (see 4.5.1). For fixed format sense data, the RESPONSE CODE field shall be set to 70h or 71h.

The meaning of the FILEMARK bit is device-type or command specific (e.g., see the SSC-3 READ command and SPACE command for examples of FILEMARK bit usage) and the bit is defined in a command standard (see 3.1.27).

The meaning of the end-of-medium (EOM) bit is device-type or command specific (e.g., see the SSC-3 READ command, SPACE command, and WRITE command for examples of EOM bit usage) and the bit is defined in a command standard (see 3.1.27).

The meaning of the incorrect length indicator (ILI) bit is device-type or command specific (e.g., see the SBC-3 READ LONG command, SBC-3 WRITE LONG command, and SSC-3 READ command for examples of ILI bit usage) and the bit is defined in a command standard (see 3.1.27).

The SENSE KEY, ADDITIONAL SENSE CODE, and ADDITIONAL SENSE CODE QUALIFIER fields are described in 4.5.2.1.

The contents of the INFORMATION field are device-type or command specific and are defined in a command standard (see 3.1.27).

The ADDITIONAL SENSE LENGTH field indicates the number of additional sense bytes that follow. The additional sense length shall be less than or equal to 244 (i.e., limiting the total length of the sense data to 252 bytes). If the sense data is being returned as parameter data by a REQUEST SENSE command, then the relationship between the ADDITIONAL SENSE LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The COMMAND-SPECIFIC INFORMATION field contains information that depends on the command on which the exception condition occurred.

The FIELD REPLACEABLE UNIT CODE field is described in 4.5.2.5.

A sense-key specific valid (SKSV) bit set to one indicates the SENSE KEY SPECIFIC field contains valid information as defined in this standard. An SKSV bit set to zero indicates that the SENSE KEY SPECIFIC field is not as defined by this standard.

The SENSE KEY SPECIFIC field is described in 4.5.2.4.

The additional sense bytes may contain vendor specific data that further defines the nature of the exception condition.

#### 4.5.4 Current errors

Response codes 70h and 72h (current error) indicate that the sense data returned is the result of an error or exception condition on the task that returned the CHECK CONDITION status or a protocol specific failure condition. This includes errors generated during processing of the command. It also includes errors not related to any command that are detected during processing of a command (e.g., disk servo-mechanism failure, off-track errors, or power-up test errors).

#### 4.5.5 Deferred errors

Response codes 71h and 73h (deferred error) indicate that the sense data returned is the result of an error or exception condition that occurred during processing of a previous command for which GOOD status or CONDITION MET status has already been returned. Such commands are associated with the use of the immediate bit and with some forms of caching. Device servers that implement these features shall implement deferred error reporting.

The deferred error may be indicated by returning CHECK CONDITION status to an application client accessed through a defined I\_T nexus as described in this subclause.

If the task terminates with CHECK CONDITION status and the sense data describes a deferred error, the command for the terminated task shall not have been processed. After the device server detects a deferred error condition, it shall return a deferred error according to the following rules:

- a) If no external intervention is necessary to recover a deferred error, a deferred error indication shall not be returned unless required by the error handling parameters of a MODE SELECT command. The occurrence of the error may be logged;
- b) If it is possible to associate a deferred error with an I\_T nexus and with a particular function or a particular subset of data, and the error is either unrecovered or required to be reported by the mode parameters, then a deferred error indication shall be returned for a command received on the I\_T nexus associated with the deferred error. If an application client request received on an I\_T nexus other than the I\_T nexus associated with the deferred error attempts to access the particular function or subset of data associated with the deferred error and the TST field equals 000b (see 7.4.6), then the device server shall complete the command with BUSY status or ACA ACTIVE status according to the requirements in SAM-4. If an application client request received on an I\_T nexus other than the I\_T nexus associated with the deferred error attempts to access the particular function or subset of data associated with the deferred error and the TST



field equals 001b, then the command attempting the access shall not be blocked by the deferred error and the cause of the deferred error may result in an error being reported for the command attempting the access;

- c) If the device server is unable to associate a deferred error with an I\_T nexus or with a particular subset of data, the device server shall return a deferred error for one command received on each I\_T nexus. If multiple deferred errors have accumulated for an I\_T nexus, only the last error shall be returned;
- d) If the SCSI target device is unable to associate a deferred error with a particular logical unit, it shall establish a deferred error for every logical unit and shall return the deferred error for one command for each logical unit received on each appropriate I\_T nexus; or
- e) If a task has never entered the enabled task state, and a deferred error occurs, the task shall be terminated with CHECK CONDITION status and deferred error information returned in the sense data. If a deferred error occurs after a task has entered the enabled task state and the task is affected by the error, the task shall be terminated with CHECK CONDITION status and the current error information shall be returned in the sense data. In this case, if the current error information does not adequately define the deferred error condition, a deferred error may be returned after the current error information has been returned. If a deferred error occurs after a task has entered the enabled task state and the task completes successfully, the device server may choose to return the deferred error information after the completion of the current command in conjunction with a subsequent command that has not begun processing.

NOTE 7 - A deferred error may indicate that an operation was unsuccessful long after GOOD status was returned. If the application client is unable to replicate or recover from other sources the data that is being written using cached or buffered write operations, then synchronization commands should be performed before the critical data is destroyed. This is necessary for actions taken when deferred errors occur in the storing of the data. The synchronizing process should provide the necessary commands to allow returning CHECK CONDITION status and subsequent returning of deferred error sense information after all cached or buffered operations are completed.

#### 4.5.6 Sense key and additional sense code definitions

The sense keys are defined in table 40.

**Table 40 — Sense key descriptions (part 1 of 2)**

Sense Key	Description
0h	<b>NO SENSE:</b> Indicates that there is no specific sense key information to be reported. This may occur for a successful command or for a command that receives CHECK CONDITION status because one of the FILEMARK, EOM, or ILI bits is set to one.
1h	<b>RECOVERED ERROR:</b> Indicates that the command completed successfully, with some recovery action performed by the device server. Details may be determined by examining the additional sense bytes and the INFORMATION field. When multiple recovered errors occur during one command, the choice of which error to report (e.g., first, last, most severe) is vendor specific.
2h	<b>NOT READY:</b> Indicates that the logical unit is not accessible. Operator intervention may be required to correct this condition.
3h	<b>MEDIUM ERROR:</b> Indicates that the command terminated with a non-recovered error condition that may have been caused by a flaw in the medium or an error in the recorded data. This sense key may also be returned if the device server is unable to distinguish between a flaw in the medium and a specific hardware failure (i.e., sense key 4h).
4h	<b>HARDWARE ERROR:</b> Indicates that the device server detected a non-recoverable hardware failure (e.g., controller failure, device failure, or parity error) while performing the command or during a self test.

Table 40 — Sense key descriptions (part 2 of 2)

Sense Key	Description
5h	<p><b>ILLEGAL REQUEST:</b> Indicates that:</p> <ul style="list-style-type: none"> <li>a) The command was addressed to an incorrect logical unit number (see SAM-4);</li> <li>b) The command had an invalid task attribute (see SAM-4);</li> <li>c) The command was addressed to a logical unit whose current configuration prohibits processing the command;</li> <li>d) There was an illegal parameter in the CDB; or</li> <li>e) There was an illegal parameter in the additional parameters supplied as data for some commands (e.g., PERSISTENT RESERVE OUT).</li> </ul> <p>If the device server detects an invalid parameter in the CDB, it shall terminate the command without altering the medium. If the device server detects an invalid parameter in the additional parameters supplied as data, the device server may have already altered the medium.</p>
6h	<b>UNIT ATTENTION:</b> Indicates that a unit attention condition has been established (e.g., the removable medium may have been changed, a logical unit reset occurred). See SAM-4.
7h	<b>DATA PROTECT:</b> Indicates that a command that reads or writes the medium was attempted on a block that is protected. The read or write operation is not performed.
8h	<b>BLANK CHECK:</b> Indicates that a write-once device or a sequential-access device encountered blank medium or format-defined end-of-data indication while reading or that a write-once device encountered a non-blank medium while writing.
9h	<b>VENDOR SPECIFIC:</b> This sense key is available for reporting vendor specific conditions.
Ah	<b>COPY ABORTED:</b> Indicates an EXTENDED COPY command was aborted due to an error condition on the source device, the destination device, or both (see 6.3.3).
Bh	<b>ABORTED COMMAND:</b> Indicates that the device server aborted the command. The application client may be able to recover by trying the command again.
Ch	Obsolete
Dh	<b>VOLUME OVERFLOW:</b> Indicates that a buffered SCSI device has reached the end-of-partition and data may remain in the buffer that has not been written to the medium. One or more RECOVER BUFFERED DATA command(s) may be issued to read the unwritten data from the buffer. (See SSC-2.)
Eh	<b>MISCOMPARE:</b> Indicates that the source data did not match the data read from the medium.
Fh	Reserved

The additional sense codes (i.e., the ADDITIONAL SENSE CODE field and ADDITIONAL SENSE CODE QUALIFIER field values returned in sense data) are defined in table 41.

**Table 41 — ASC and ASCQ assignments (part 1 of 16)**

D – Direct Access Block Device (SBC-3)														Device Column key	
. T – Sequential Access Device (SSC-3)														blank = code not used	
. L – Printer Device (SSC)														not blank = code used	
. P – Processor Device (SPC-2)															
. . W – Write Once Block Device (SBC)															
. . . R – C/DVD Device (MMC-6)															
. . . O – Optical Memory Block Device (SBC)															
. . . . M – Media Changer Device (SMC-3)															
. . . . A – Storage Array Device (SCC-2)															
. . . . E – SCSI Enclosure Services device (SES)															
. . . . . B – Simplified Direct-Access (Reduced Block) device (RBC)															
. . . . . K – Optical Card Reader/Writer device (OCRW)															
. . . . . V – Automation/Device Interface device (ADC)															
. . . . . F – Object-based Storage Device (OSD)															
ASC	ASCQ	DT	L	P	W	R	O	M	A	E	B	K	V	F	Description
20h	0Bh	DT													ACCESS DENIED - ACL LUN CONFLICT
20h	08h	DT													ACCESS DENIED - ENROLLMENT CONFLICT
20h	01h	DT													ACCESS DENIED - INITIATOR PENDING-ENROLLED
20h	09h	DT													ACCESS DENIED - INVALID LU IDENTIFIER
20h	03h	DT													ACCESS DENIED - INVALID MGMT ID KEY
20h	0Ah	DT													ACCESS DENIED - INVALID PROXY TOKEN
20h	02h	DT													ACCESS DENIED - NO ACCESS RIGHTS
4Bh	03h	DT													ACK/NAK TIMEOUT
67h	02h								A						ADD LOGICAL UNIT FAILED
13h	00h	D			W		O						BK		ADDRESS MARK NOT FOUND FOR DATA FIELD
12h	00h	D			W		O						BK		ADDRESS MARK NOT FOUND FOR ID FIELD
67h	08h								A						ASSIGN FAILURE OCCURRED
27h	03h		T					R							ASSOCIATED WRITE PROTECT
2Ah	06h	DT	L	P	W	R	O	M	A	E	B	K	V	F	ASYMMETRIC ACCESS STATE CHANGED
47h	04h	DT	L	P	W	R	O	M	A	E	B	K	V	F	ASYNCHRONOUS INFORMATION PROTECTION ERROR DETECTED
44h	71h	DT											B		ATA DEVICE FAILED SET FEATURES
67h	0Bh	DT											B		ATA DEVICE FEATURE NOT ENABLED
00h	1Dh	DT											B		ATA PASS THROUGH INFORMATION AVAILABLE
67h	06h								A						ATTACHMENT OF LOGICAL UNIT FAILED
00h	11h							R							AUDIO PLAY OPERATION IN PROGRESS
00h	12h							R							AUDIO PLAY OPERATION PAUSED
00h	14h							R							AUDIO PLAY OPERATION STOPPED DUE TO ERROR
00h	13h							R							AUDIO PLAY OPERATION SUCCESSFULLY COMPLETED
74h	40h	DT				R			M	A	E	B	K	V	AUTHENTICATION FAILED
66h	00h														AUTOMATIC DOCUMENT FEEDER COVER UP
66h	01h														AUTOMATIC DOCUMENT FEEDER LIFT UP
55h	06h	DT			W	R	O	M					B		AUXILIARY MEMORY OUT OF SPACE
11h	12h	DT			W	R	O	M					B		AUXILIARY MEMORY READ ERROR
0Ch	0Bh	DT			W	R	O	M					B		AUXILIARY MEMORY WRITE ERROR
00h	04h		T												BEGINNING-OF-PARTITION/MEDIUM DETECTED
0Ch	06h	DT		W		O							B		BLOCK NOT COMPRESSIBLE
14h	04h		T												BLOCK SEQUENCE ERROR
29h	03h	DT	L	P	W	R	O	M	A	E	B	K	V	F	BUS DEVICE RESET FUNCTION OCCURRED
11h	0Eh	DT			W	R	O						B		CANNOT DECOMPRESS USING DECLARED ALGORITHM
30h	06h	DT			W	R	O						B		CANNOT FORMAT MEDIUM - INCOMPATIBLE MEDIUM
30h	02h	DT			W	R	O						BK		CANNOT READ MEDIUM - INCOMPATIBLE FORMAT

Annex D contains the ASC and ASCQ assignments in numeric order.

Table 41 — ASC and ASCQ assignments (part 2 of 16)

D – Direct Access Block Device (SBC-3)														Device Column key	
. T – Sequential Access Device (SSC-3)														blank = code not used	
. L – Printer Device (SSC)														not blank = code used	
. P – Processor Device (SPC-2)															
. W – Write Once Block Device (SBC)															
. R – C/DVD Device (MMC-6)															
. O – Optical Memory Block Device (SBC)															
. M – Media Changer Device (SMC-3)															
. A – Storage Array Device (SCC-2)															
. E – SCSI Enclosure Services device (SES)															
. B – Simplified Direct-Access (Reduced Block) device (RBC)															
. K – Optical Card Reader/Writer device (OCRW)															
. V – Automation/Device Interface device (ADC)															
. F – Object-based Storage Device (OSD)															
ASC	ASCQ	DT	L	P	W	R	O	M	A	E	B	K	V	F	Description
30h	01h	DT			W	R	O				B	K			CANNOT READ MEDIUM - UNKNOWN FORMAT
30h	08h										R				CANNOT WRITE - APPLICATION CODE MISMATCH
30h	05h	DT			W	R	O				B	K			CANNOT WRITE MEDIUM - INCOMPATIBLE FORMAT
30h	04h	DT			W	R	O				B	K			CANNOT WRITE MEDIUM - UNKNOWN FORMAT
2Ah	09h	D													CAPACITY DATA HAS CHANGED
52h	00h		T												CARTRIDGE FAULT
73h	00h					R									CD CONTROL ERROR
24h	01h	DT	L	P	W	R	O			A	E	B	K	V	CDB DECRYPTION ERROR
3Fh	02h	DT	L	P	W	R	O			M			B	K	CHANGED OPERATING DEFINITION
11h	06h					W	R	O			B				CIRC UNRECOVERED ERROR
30h	03h	DT				R							K		CLEANING CARTRIDGE INSTALLED
30h	07h	DT	L				W	R	O	M	A	E	B	K	CLEANING FAILURE
30h	0Ah	DT			W	R	O			A	E	B	K		CLEANING REQUEST REJECTED
00h	17h	DT	L				W	R	O	M	A	E	B	K	CLEANING REQUESTED
4Ah	00h	DT	L	P	W	R	O			M	A	E	B	K	COMMAND PHASE ERROR
2Ch	00h	DT	L	P	W	R	O			M	A	E	B	K	COMMAND SEQUENCE ERROR
6Eh	00h									A					COMMAND TO LOGICAL UNIT FAILED
2Fh	00h	DT	L	P	W	R	O			M	A	E	B	K	COMMANDS CLEARED BY ANOTHER INITIATOR
2Fh	02h	DT	L	P	W	R	O			M	A	E	B	K	COMMANDS CLEARED BY DEVICE SERVER
2Fh	01h	D													COMMANDS CLEARED BY POWER LOSS NOTIFICATION
3Fh	04h	DT				W	R	O	M	A	E	B	K		COMPONENT DEVICE ATTACHED
0Ch	04h	DT			W		O					B			COMPRESSION CHECK MISCOMPARE ERROR
27h	06h					R								F	CONDITIONAL WRITE PROTECT
67h	00h									A					CONFIGURATION FAILURE
67h	01h									A					CONFIGURATION OF INCAPABLE LOGICAL UNITS FAILED
6Fh	07h					R									CONFLICT IN BINDING NONCE RECORDING
00h	1Eh	DT				R			M	A	E	B	K	V	CONFLICTING SA CREATION REQUEST
5Dh	25h	D										B			CONTROLLER IMPENDING FAILURE ACCESS TIMES TOO HIGH
5Dh	27h	D										B			CONTROLLER IMPENDING FAILURE CHANNEL PARAMETRICS
5Dh	28h	D										B			CONTROLLER IMPENDING FAILURE CONTROLLER DETECTED
5Dh	22h	D										B			CONTROLLER IMPENDING FAILURE DATA ERROR RATE TOO HIGH
5Dh	2Ch	D										B			CONTROLLER IMPENDING FAILURE DRIVE CALIBRATION RETRY COUNT
5Dh	21h	D										B			CONTROLLER IMPENDING FAILURE DRIVE ERROR RATE TOO HIGH
5Dh	20h	D										B			CONTROLLER IMPENDING FAILURE GENERAL HARD DRIVE FAILURE
5Dh	23h	D										B			CONTROLLER IMPENDING FAILURE SEEK ERROR RATE TOO HIGH
5Dh	2Ah	D										B			CONTROLLER IMPENDING FAILURE SEEK TIME PERFORMANCE
5Dh	2Bh	D										B			CONTROLLER IMPENDING FAILURE SPIN-UP RETRY COUNT
5Dh	26h	D										B			CONTROLLER IMPENDING FAILURE START UNIT TIMES TOO HIGH
5Dh	29h	D										B			CONTROLLER IMPENDING FAILURE THROUGHPUT PERFORMANCE
5Dh	24h	D										B			CONTROLLER IMPENDING FAILURE TOO MANY BLOCK REASSIGNS
Annex D contains the ASC and ASCQ assignments in numeric order.															

Annex D contains the ASC and ASCQ assignments in numeric order.

**Table 41 — ASC and ASCQ assignments** (part 3 of 16)

D – Direct Access Block Device (SBC-3)					Device Column key
. T – Sequential Access Device (SSC-3)					blank = code not used
. L – Printer Device (SSC)					not blank = code used
. P – Processor Device (SPC-2)					
. W – Write Once Block Device (SBC)					
. R – C/DVD Device (MMC-6)					
. O – Optical Memory Block Device (SBC)					
. M – Media Changer Device (SMC-3)					
. A – Storage Array Device (SCC-2)					
. E – SCSI Enclosure Services device (SES)					
. B – Simplified Direct-Access (Reduced Block) device (RBC)					
. K – Optical Card Reader/Writer device (OCRW)					
. V – Automation/Device Interface device (ADC)					
. F – Object-based Storage Device (OSD)					
ASC	ASCQ	DTLPWROMAEBKVF	Description		
2Bh	00h	DTLPWRO	K	COPY CANNOT EXECUTE SINCE HOST CANNOT DISCONNECT	
6Fh	00h	R		COPY PROTECTION KEY EXCHANGE FAILURE - AUTHENTICATION FAILURE	
6Fh	02h	R		COPY PROTECTION KEY EXCHANGE FAILURE - KEY NOT ESTABLISHED	
6Fh	01h	R		COPY PROTECTION KEY EXCHANGE FAILURE - KEY NOT PRESENT	
26h	0Dh	DTLPWRO	K	COPY SEGMENT GRANULARITY VIOLATION	
0Dh	05h	DTLPWRO	A K	COPY TARGET DEVICE DATA OVERRUN	
0Dh	04h	DTLPWRO	A K	COPY TARGET DEVICE DATA UNDERRUN	
0Dh	02h	DTLPWRO	A K	COPY TARGET DEVICE NOT REACHABLE	
67h	07h	A		CREATION OF LOGICAL UNIT FAILED	
74h	04h	T		CRYPTOGRAPHIC INTEGRITY VALIDATION FAILED	
73h	10h	R		CURRENT POWER CALIBRATION AREA ALMOST FULL	
73h	11h	R		CURRENT POWER CALIBRATION AREA IS FULL	
2Ch	04h	R		CURRENT PROGRAM AREA IS EMPTY	
2Ch	03h	R		CURRENT PROGRAM AREA IS NOT EMPTY	
30h	09h	R		CURRENT SESSION NOT FIXATED FOR APPEND	
5Dh	35h	D	B	DATA CHANNEL IMPENDING FAILURE ACCESS TIMES TOO HIGH	
5Dh	37h	D	B	DATA CHANNEL IMPENDING FAILURE CHANNEL PARAMETRICS	
5Dh	38h	D	B	DATA CHANNEL IMPENDING FAILURE CONTROLLER DETECTED	
5Dh	32h	D	B	DATA CHANNEL IMPENDING FAILURE DATA ERROR RATE TOO HIGH	
5Dh	3Ch	D	B	DATA CHANNEL IMPENDING FAILURE DRIVE CALIBRATION RETRY COUNT	
5Dh	31h	D	B	DATA CHANNEL IMPENDING FAILURE DRIVE ERROR RATE TOO HIGH	
5Dh	30h	D	B	DATA CHANNEL IMPENDING FAILURE GENERAL HARD DRIVE FAILURE	
5Dh	33h	D	B	DATA CHANNEL IMPENDING FAILURE SEEK ERROR RATE TOO HIGH	
5Dh	3Ah	D	B	DATA CHANNEL IMPENDING FAILURE SEEK TIME PERFORMANCE	
5Dh	3Bh	D	B	DATA CHANNEL IMPENDING FAILURE SPIN-UP RETRY COUNT	
5Dh	36h	D	B	DATA CHANNEL IMPENDING FAILURE START UNIT TIMES TOO HIGH	
5Dh	39h	D	B	DATA CHANNEL IMPENDING FAILURE THROUGHPUT PERFORMANCE	
5Dh	34h	D	B	DATA CHANNEL IMPENDING FAILURE TOO MANY BLOCK REASSIGNS	
55h	0Ah	M		DATA CURRENTLY UNAVAILABLE	
26h	05h	DTLPWRO	A BK	DATA DECRYPTION ERROR	
26h	10h	T		DATA DECRYPTION KEY FAIL LIMIT REACHED	
2Ah	0Dh	T		DATA ENCRYPTION CAPABILITIES CHANGED	
74h	21h	T		DATA ENCRYPTION CONFIGURATION PREVENTED	
2Ah	13h	T		DATA ENCRYPTION KEY INSTANCE COUNTER HAS CHANGED	
2Ah	11h	T		DATA ENCRYPTION PARAMETERS CHANGED BY ANOTHER I_T NEXUS	
2Ah	12h	T		DATA ENCRYPTION PARAMETERS CHANGED BY VENDOR SPECIFIC EVENT	
0Ch	05h	DT	W O B	DATA EXPANSION OCCURRED DURING COMPRESSION	
69h	00h	A		DATA LOSS ON LOGICAL UNIT	
4Bh	05h	DT PWROMAEBK		DATA OFFSET ERROR	
41h	00h	D		DATA PATH FAILURE (SHOULD USE 40 NN)	

Annex D contains the ASC and ASCQ assignments in numeric order.

Table 41 — ASC and ASCQ assignments (part 4 of 16)

D – Direct Access Block Device (SBC-3)										Device Column key
. T – Sequential Access Device (SSC-3)										blank = code not used
. L – Printer Device (SSC)										not blank = code used
. P – Processor Device (SPC-2)										
. W – Write Once Block Device (SBC)										
. R – C/DVD Device (MMC-6)										
. O – Optical Memory Block Device (SBC)										
. M – Media Changer Device (SMC-3)										
. A – Storage Array Device (SCC-2)										
. E – SCSI Enclosure Services device (SES)										
. B – Simplified Direct-Access (Reduced Block) device (RBC)										
. K – Optical Card Reader/Writer device (OCRW)										
. V – Automation/Device Interface device (ADC)										
. F – Object-based Storage Device (OSD)										
ASC	ASCQ	DTLPWROMAEBKVF	Description							
47h	01h	DTLPWROMAEBKVF	DATA PHASE CRC ERROR DETECTED							
4Bh	00h	DTLPWROMAEBKVF	DATA PHASE ERROR							
11h	07h	W O B	DATA RE-SYNCHRONIZATION ERROR							
16h	03h	D W O BK	DATA SYNC ERROR - DATA AUTO-REALLOCATED							
16h	01h	D W O BK	DATA SYNC ERROR - DATA REWRITTEN							
16h	04h	D W O BK	DATA SYNC ERROR - RECOMMEND REASSIGNMENT							
16h	02h	D W O BK	DATA SYNC ERROR - RECOMMEND REWRITE							
16h	00h	D W O BK	DATA SYNCHRONIZATION MARK ERROR							
3Bh	1Bh	M	DATA TRANSFER DEVICE INSERTED							
3Bh	1Ah	M	DATA TRANSFER DEVICE REMOVED							
11h	0Dh	DT WRO B	DE-COMPRESSION CRC ERROR							
71h	00h	T	DECOMPRESSION EXCEPTION LONG ALGORITHM ID							
70h	NNh	T	DECOMPRESSION EXCEPTION SHORT ALGORITHM ID OF NN							
19h	00h	D O K	DEFECT LIST ERROR							
19h	03h	D O K	DEFECT LIST ERROR IN GROWN LIST							
19h	02h	D O K	DEFECT LIST ERROR IN PRIMARY LIST							
19h	01h	D O K	DEFECT LIST NOT AVAILABLE							
1Ch	00h	D O BK	DEFECT LIST NOT FOUND							
32h	01h	D W O BK	DEFECT LIST UPDATE FAILURE							
0Ch	0Fh	R	DEFECTS IN ERROR WINDOW							
3Fh	05h	DT WROMAEBK	DEVICE IDENTIFIER CHANGED							
29h	04h	DTLPWROMAEBKVF	DEVICE INTERNAL RESET							
40h	NNh	DTLPWROMAEBKVF	DIAGNOSTIC FAILURE ON COMPONENT NN (80h-FFh)							
74h	08h	DT R M E VF	DIGITAL SIGNATURE VALIDATION FAILURE							
66h	02h		DOCUMENT JAM IN AUTOMATIC DOCUMENT FEEDER							
66h	03h		DOCUMENT MISS FEED AUTOMATIC IN DOCUMENT FEEDER							
6Fh	05h	R	DRIVE REGION MUST BE PERMANENT/REGION RESET COUNT ERROR							
3Fh	0Fh	DTLPWROMAEBKVF	ECHO BUFFER OVERWRITTEN							
3Bh	18h	M	ELEMENT DISABLED							
3Bh	19h	M	ELEMENT ENABLED							
72h	04h	R	EMPTY OR PARTIALLY WRITTEN RESERVED TRACK							
34h	00h	DTLPWROMAEBKVF	ENCLOSURE FAILURE							
35h	05h	DTL WROMAEBKVF	ENCLOSURE SERVICES CHECKSUM ERROR							
35h	00h	DTLPWROMAEBKVF	ENCLOSURE SERVICES FAILURE							
35h	03h	DTLPWROMAEBKVF	ENCLOSURE SERVICES TRANSFER FAILURE							
35h	04h	DTLPWROMAEBKVF	ENCLOSURE SERVICES TRANSFER REFUSED							
35h	02h	DTLPWROMAEBKVF	ENCLOSURE SERVICES UNAVAILABLE							
74h	0Ah	T	ENCRYPTED BLOCK NOT RAW READ ENABLED							
74h	0Dh	T	ENCRYPTION ALGORITHM DISABLED							
74h	09h	T	ENCRYPTION MODE MISMATCH ON READ							
Annex D contains the ASC and ASCQ assignments in numeric order.										

Table 41 — ASC and ASCQ assignments (part 5 of 16)

D – Direct Access Block Device (SBC-3)										Device Column key
. T – Sequential Access Device (SSC-3)										blank = code not used
. L – Printer Device (SSC)										not blank = code used
. P – Processor Device (SPC-2)										
. W – Write Once Block Device (SBC)										
. R – C/DVD Device (MMC-6)										
. O – Optical Memory Block Device (SBC)										
. M – Media Changer Device (SMC-3)										
. A – Storage Array Device (SCC-2)										
. E – SCSI Enclosure Services device (SES)										
. B – Simplified Direct-Access (Reduced Block) device (RBC)										
. K – Optical Card Reader/Writer device (OCRW)										
. V – Automation/Device Interface device (ADC)										
. F – Object-based Storage Device (OSD)										
ASC	ASCQ	DTLPWROMAEBKVF	Description							
74h	07h	T	ENCRYPTION PARAMETERS NOT USEABLE							
3Bh	0Fh		R	END OF MEDIUM REACHED						
63h	00h		R	END OF USER AREA ENCOUNTERED ON THIS TRACK						
00h	05h	T L	END-OF-DATA DETECTED							
14h	03h	T	END-OF-DATA NOT FOUND							
00h	02h	T	END-OF-PARTITION/MEDIUM DETECTED							
51h	00h	T	RO	ERASE FAILURE						
51h	01h		R	ERASE FAILURE - INCOMPLETE ERASE OPERATION DETECTED						
00h	18h	T	ERASE OPERATION IN PROGRESS							
74h	05h	T	ERROR DECRYPTING DATA							
0Dh	00h	DTLPWRO	A	K	ERROR DETECTED BY THIRD PARTY TEMPORARY INITIATOR					
2Ah	0Ah	DT	ERROR HISTORY I_T NEXUS CLEARED							
2Ah	0Bh	DT	ERROR HISTORY SNAPSHOT RELEASED							
0Ah	00h	DTLPWROMAEBKVF	ERROR LOG OVERFLOW							
11h	10h		R	ERROR READING ISRC NUMBER						
11h	0Fh		R	ERROR READING UPC/EAN NUMBER						
2Ah	0Ch			F	ERROR RECOVERY ATTRIBUTES HAVE CHANGED					
11h	02h	DT	WRO	BK	ERROR TOO LONG TO CORRECT					
38h	06h			B	ESN - DEVICE BUSY CLASS EVENT					
38h	04h			B	ESN - MEDIA CLASS EVENT					
38h	02h			B	ESN - POWER MANAGEMENT CLASS EVENT					
38h	00h			B	EVENT STATUS NOTIFICATION					
03h	02h	T	EXCESSIVE WRITE ERRORS							
67h	04h		A	EXCHANGE OF LOGICAL UNIT FAILED						
74h	6Fh	T	EXTERNAL DATA ENCRYPTION CONTROL ERROR							
74h	6Eh	T	EXTERNAL DATA ENCRYPTION CONTROL TIMEOUT							
74h	61h			V	EXTERNAL DATA ENCRYPTION KEY MANAGER ACCESS ERROR					
74h	62h			V	EXTERNAL DATA ENCRYPTION KEY MANAGER ERROR					
74h	63h			V	EXTERNAL DATA ENCRYPTION KEY NOT FOUND					
74h	64h			V	EXTERNAL DATA ENCRYPTION REQUEST NOT AUTHORIZED					
3Bh	07h		L	FAILED TO SENSE BOTTOM-OF-FORM						
3Bh	06h		L	FAILED TO SENSE TOP-OF-FORM						
5Dh	00h	DTLPWROMAEBKVF	FAILURE PREDICTION THRESHOLD EXCEEDED							
5Dh	FFh	DTLPWROMAEBKVF	FAILURE PREDICTION THRESHOLD EXCEEDED (FALSE)							
00h	01h	T	FILEMARK DETECTED							
14h	02h	T	FILEMARK OR SETMARK NOT FOUND							
5Dh	65h	D		B	FIRMWARE IMPENDING FAILURE ACCESS TIMES TOO HIGH					
5Dh	67h	D		B	FIRMWARE IMPENDING FAILURE CHANNEL PARAMETRICS					
5Dh	68h	D		B	FIRMWARE IMPENDING FAILURE CONTROLLER DETECTED					
5Dh	62h	D		B	FIRMWARE IMPENDING FAILURE DATA ERROR RATE TOO HIGH					
Annex D contains the ASC and ASCQ assignments in numeric order.										

Table 41 — ASC and ASCQ assignments (part 6 of 16)

D – Direct Access Block Device (SBC-3)										Device Column key
. T – Sequential Access Device (SSC-3)										blank = code not used
. L – Printer Device (SSC)										not blank = code used
. P – Processor Device (SPC-2)										
. . W – Write Once Block Device (SBC)										
. . R – C/DVD Device (MMC-6)										
. . O – Optical Memory Block Device (SBC)										
. . . M – Media Changer Device (SMC-3)										
. . . A – Storage Array Device (SCC-2)										
. . . E – SCSI Enclosure Services device (SES)										
. . . B – Simplified Direct-Access (Reduced Block) device (RBC)										
. . . K – Optical Card Reader/Writer device (OCRW)										
. . . V – Automation/Device Interface device (ADC)										
. . . F – Object-based Storage Device (OSD)										
ASC	ASCQ	DTLPWROMAEBKVF	Description							
5Dh	6Ch	D				B	FIRMWARE IMPENDING FAILURE DRIVE CALIBRATION RETRY COUNT			
5Dh	61h	D				B	FIRMWARE IMPENDING FAILURE DRIVE ERROR RATE TOO HIGH			
5Dh	60h	D				B	FIRMWARE IMPENDING FAILURE GENERAL HARD DRIVE FAILURE			
5Dh	63h	D				B	FIRMWARE IMPENDING FAILURE SEEK ERROR RATE TOO HIGH			
5Dh	6Ah	D				B	FIRMWARE IMPENDING FAILURE SEEK TIME PERFORMANCE			
5Dh	6Bh	D				B	FIRMWARE IMPENDING FAILURE SPIN-UP RETRY COUNT			
5Dh	66h	D				B	FIRMWARE IMPENDING FAILURE START UNIT TIMES TOO HIGH			
5Dh	69h	D				B	FIRMWARE IMPENDING FAILURE THROUGHPUT PERFORMANCE			
5Dh	64h	D				B	FIRMWARE IMPENDING FAILURE TOO MANY BLOCK REASSIGNS			
09h	02h			WRO			K	FOCUS SERVO FAILURE		
31h	01h	D	L		RO		B	FORMAT COMMAND FAILED		
28h	02h				R			FORMAT-LAYER MAY HAVE CHANGED		
58h	00h				O			GENERATION DOES NOT EXIST		
1Ch	02h	D			O		BK	GROWN DEFECT LIST NOT FOUND		
5Dh	15h	D				B	HARDWARE IMPENDING FAILURE ACCESS TIMES TOO HIGH			
5Dh	17h	D				B	HARDWARE IMPENDING FAILURE CHANNEL PARAMETRICS			
5Dh	18h	D				B	HARDWARE IMPENDING FAILURE CONTROLLER DETECTED			
5Dh	12h	D				B	HARDWARE IMPENDING FAILURE DATA ERROR RATE TOO HIGH			
5Dh	1Ch	D				B	HARDWARE IMPENDING FAILURE DRIVE CALIBRATION RETRY COUNT			
5Dh	11h	D				B	HARDWARE IMPENDING FAILURE DRIVE ERROR RATE TOO HIGH			
5Dh	10h	D				B	HARDWARE IMPENDING FAILURE GENERAL HARD DRIVE FAILURE			
5Dh	13h	D				B	HARDWARE IMPENDING FAILURE SEEK ERROR RATE TOO HIGH			
5Dh	1Ah	D				B	HARDWARE IMPENDING FAILURE SEEK TIME PERFORMANCE			
5Dh	1Bh	D				B	HARDWARE IMPENDING FAILURE SPIN-UP RETRY COUNT			
5Dh	16h	D				B	HARDWARE IMPENDING FAILURE START UNIT TIMES TOO HIGH			
5Dh	19h	D				B	HARDWARE IMPENDING FAILURE THROUGHPUT PERFORMANCE			
5Dh	14h	D				B	HARDWARE IMPENDING FAILURE TOO MANY BLOCK REASSIGNS			
27h	01h	DT		WRO			BK	HARDWARE WRITE PROTECTED		
09h	04h	DT		WRO			B	HEAD SELECT FAULT		
00h	06h	DTLPWROMAEBKVF						I/O PROCESS TERMINATED		
10h	00h	D		W	O		BK	ID CRC OR ECC ERROR		
5Eh	03h	DTLPWRO		A			K	IDLE CONDITION ACTIVATED BY COMMAND		
5Eh	01h	DTLPWRO		A			K	IDLE CONDITION ACTIVATED BY TIMER		
20h	06h		T					ILLEGAL COMMAND WHILE IN EXPLICIT ADDRESS MODE		
20h	07h		T					ILLEGAL COMMAND WHILE IN IMPLICIT ADDRESS MODE		
20h	04h		T					ILLEGAL COMMAND WHILE IN WRITE CAPABLE STATE		
22h	00h	D						ILLEGAL FUNCTION (USE 20 00, 24 00, OR 26 00)		
64h	00h				R			ILLEGAL MODE FOR THIS TRACK		
2Ch	05h					B		ILLEGAL POWER CONDITION REQUEST		
2Ah	07h	DTLPWROMAEBKVF						IMPLICIT ASYMMETRIC ACCESS STATE TRANSITION FAILED		

Annex D contains the ASC and ASCQ assignments in numeric order.

Annex D contains the ASC and ASCQ assignments in numeric order.



Table 41 — ASC and ASCQ assignments (part 7 of 16)

D – Direct Access Block Device (SBC-3)						Device Column key
. T – Sequential Access Device (SSC-3)						blank = code not used
. L – Printer Device (SSC)						not blank = code used
. P – Processor Device (SPC-2)						
. W – Write Once Block Device (SBC)						
. R – C/DVD Device (MMC-6)						
. O – Optical Memory Block Device (SBC)						
. M – Media Changer Device (SMC-3)						
. A – Storage Array Device (SCC-2)						
. E – SCSI Enclosure Services device (SES)						
. B – Simplified Direct-Access (Reduced Block) device (RBC)						
. K – Optical Card Reader/Writer device (OCRW)						
. V – Automation/Device Interface device (ADC)						
. F – Object-based Storage Device (OSD)						
ASC	ASCQ	DTLPWROMAEBKVF	Description			
28h	01h	DT WROM B	IMPORT OR EXPORT ELEMENT ACCESSED			
28h	03h	M	IMPORT/EXPORT ELEMENT ACCESSED, MEDIUM CHANGED			
30h	00h	DT WROM BK	INCOMPATIBLE MEDIUM INSTALLED			
30h	12h	M	INCOMPATIBLE VOLUME QUALIFIER			
30h	11h	M	INCOMPATIBLE VOLUME TYPE			
11h	08h	T	INCOMPLETE BLOCK READ			
26h	11h	T	INCOMPLETE KEY-ASSOCIATED DATA SET			
0Dh	03h	DTLPWRO A K	INCORRECT COPY TARGET DEVICE TYPE			
74h	03h	T	INCORRECT DATA ENCRYPTION KEY			
74h	0Bh	T	INCORRECT ENCRYPTION PARAMETERS			
0Eh	02h	DT PWROMAEBK F	INFORMATION UNIT TOO LONG			
0Eh	01h	DT PWROMAEBK F	INFORMATION UNIT TOO SHORT			
47h	03h	DTLPWROMAEBKVF	INFORMATION UNIT iuCRC ERROR DETECTED			
6Ah	00h	A	INFORMATIONAL, REFER TO LOG			
48h	00h	DTLPWROMAEBKVF	INITIATOR DETECTED ERROR MESSAGE RECEIVED			
4Bh	06h	DT PWROMAEBK	INITIATOR RESPONSE TIMEOUT			
26h	0Bh	DTLPWRO K	INLINE DATA LENGTH EXCEEDED			
3Fh	03h	DTLPWROMAEBKVF	INQUIRY DATA HAS CHANGED			
55h	05h	DT PWROMAEBK	INSUFFICIENT ACCESS CONTROL RESOURCES			
6Fh	06h	R	INSUFFICIENT BLOCK COUNT FOR BINDING NONCE RECORDING			
55h	04h	DTLPWROMAE K	INSUFFICIENT REGISTRATION RESOURCES			
55h	02h	DTLPWROMAE K	INSUFFICIENT RESERVATION RESOURCES			
55h	03h	DTLPWROMAE K	INSUFFICIENT RESOURCES			
2Eh	00h	R	INSUFFICIENT TIME FOR OPERATION			
44h	00h	DTLPWROMAEBKVF	INTERNAL TARGET FAILURE			
21h	02h	R	INVALID ADDRESS FOR WRITE			
3Dh	00h	DTLPWROMAE K	INVALID BITS IN IDENTIFY MESSAGE			
2Ch	02h		INVALID COMBINATION OF WINDOWS SPECIFIED			
20h	00h	DTLPWROMAEBKVF	INVALID COMMAND OPERATION CODE			
26h	0Fh	F	INVALID DATA-OUT BUFFER INTEGRITY CHECK VALUE			
21h	01h	DT WROM BK	INVALID ELEMENT ADDRESS			
24h	00h	DTLPWROMAEBKVF	INVALID FIELD IN CDB			
0Eh	03h	DT P R MAEBK F	INVALID FIELD IN COMMAND INFORMATION UNIT			
26h	00h	DTLPWROMAEBKVF	INVALID FIELD IN PARAMETER LIST			
0Eh	00h	DT PWROMAEBK F	INVALID INFORMATION UNIT			
49h	00h	DTLPWROMAEBKVF	INVALID MESSAGE ERROR			
26h	0Ch	DTLPWRO K	INVALID OPERATION FOR COPY SOURCE OR DESTINATION			
64h	01h	R	INVALID PACKET SIZE			
26h	0Eh	DT PWROMAEBK	INVALID PARAMETER WHILE PORT IS ENABLED			
26h	04h	DTLPWROMAEBKVF	INVALID RELEASE OF PERSISTENT RESERVATION			
Annex D contains the ASC and ASCQ assignments in numeric order.						

Table 41 — ASC and ASCQ assignments (part 8 of 16)

D – Direct Access Block Device (SBC-3)							Device Column key
. T – Sequential Access Device (SSC-3)							blank = code not used
. L – Printer Device (SSC)							not blank = code used
. P – Processor Device (SPC-2)							
. W – Write Once Block Device (SBC)							
. R – C/DVD Device (MMC-6)							
. O – Optical Memory Block Device (SBC)							
. M – Media Changer Device (SMC-3)							
. A – Storage Array Device (SCC-2)							
. E – SCSI Enclosure Services device (SES)							
. B – Simplified Direct-Access (Reduced Block) device (RBC)							
. K – Optical Card Reader/Writer device (OCRW)							
. V – Automation/Device Interface device (ADC)							
. F – Object-based Storage Device (OSD)							
ASC	ASCQ	DTLPWROMAEBKVF	Description				
74h	12h	DT R MAEBKV	INVALID SA USAGE				
4Bh	01h	DT PWROMAEBK	INVALID TARGET PORT TRANSFER TAG RECEIVED				
21h	03h	R	INVALID WRITE CROSSING LAYER JUMP				
24h	08h	DT R MAEBKV	INVALID XCDB				
29h	07h	DTLPWROMAEBKVF	I_T NEXUS LOSS OCCURRED				
11h	05h	WRO B	L-EC UNCORRECTABLE ERROR				
60h	00h		LAMP FAILURE				
14h	07h	T	LOCATE OPERATION FAILURE				
00h	19h	T	LOCATE OPERATION IN PROGRESS				
5Bh	02h	DTLPWROM K	LOG COUNTER AT MAXIMUM				
5Bh	00h	DTLPWROM K	LOG EXCEPTION				
5Bh	03h	DTLPWROM K	LOG LIST CODES EXHAUSTED				
2Ah	02h	DTL WROMAE K	LOG PARAMETERS CHANGED				
21h	00h	DT WRO BK	LOGICAL BLOCK ADDRESS OUT OF RANGE				
10h	02h	DT W O	LOGICAL BLOCK APPLICATION TAG CHECK FAILED				
10h	01h	DT W O	LOGICAL BLOCK GUARD CHECK FAILED				
10h	03h	DT W O	LOGICAL BLOCK REFERENCE TAG CHECK FAILED				
74h	71h	DT R M E V	LOGICAL UNIT ACCESS NOT AUTHORIZED				
08h	03h	DT ROM BK	LOGICAL UNIT COMMUNICATION CRC ERROR (ULTRA-DMA/32)				
08h	00h	DTL WROMAEBKVF	LOGICAL UNIT COMMUNICATION FAILURE				
08h	02h	DTL WROMAEBKVF	LOGICAL UNIT COMMUNICATION PARITY ERROR				
08h	01h	DTL WROMAEBKVF	LOGICAL UNIT COMMUNICATION TIME-OUT				
05h	00h	DTL WROMAEBKVF	LOGICAL UNIT DOES NOT RESPOND TO SELECTION				
4Ch	00h	DTLPWROMAEBKVF	LOGICAL UNIT FAILED SELF-CONFIGURATION				
3Eh	03h	DTLPWROMAEBKVF	LOGICAL UNIT FAILED SELF-TEST				
3Eh	01h	DTLPWROMAEBKVF	LOGICAL UNIT FAILURE				
5Dh	02h	R	LOGICAL UNIT FAILURE PREDICTION THRESHOLD EXCEEDED				
3Eh	00h	DTLPWROMAEBKVF	LOGICAL UNIT HAS NOT SELF-CONFIGURED YET				
04h	01h	DTLPWROMAEBKVF	LOGICAL UNIT IS IN PROCESS OF BECOMING READY				
04h	0Ah	DTLPWROMAEBKVF	LOGICAL UNIT NOT ACCESSIBLE, ASYMMETRIC ACCESS STATE TRANSITION				
04h	0Bh	DTLPWROMAEBKVF	LOGICAL UNIT NOT ACCESSIBLE, TARGET PORT IN STANDBY STATE				
04h	0Ch	DTLPWROMAEBKVF	LOGICAL UNIT NOT ACCESSIBLE, TARGET PORT IN UNAVAILABLE STATE				
68h	00h	A	LOGICAL UNIT NOT CONFIGURED				
04h	10h	DT WROM B	LOGICAL UNIT NOT READY, AUXILIARY MEMORY NOT ACCESSIBLE				
04h	00h	DTLPWROMAEBKVF	LOGICAL UNIT NOT READY, CAUSE NOT REPORTABLE				
04h	04h	DTL RO B	LOGICAL UNIT NOT READY, FORMAT IN PROGRESS				
04h	02h	DTLPWROMAEBKVF	LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED				
04h	08h	R	LOGICAL UNIT NOT READY, LONG WRITE IN PROGRESS				
04h	03h	DTLPWROMAEBKVF	LOGICAL UNIT NOT READY, MANUAL INTERVENTION REQUIRED				
Annex D contains the ASC and ASCQ assignments in numeric order.							

Table 41 — ASC and ASCQ assignments (part 9 of 16)

D – Direct Access Block Device (SBC-3)						Device Column key
. T – Sequential Access Device (SSC-3)						blank = code not used
. L – Printer Device (SSC)						not blank = code used
. P – Processor Device (SPC-2)						
. W – Write Once Block Device (SBC)						
. R – C/DVD Device (MMC-6)						
. O – Optical Memory Block Device (SBC)						
. M – Media Changer Device (SMC-3)						
. A – Storage Array Device (SCC-2)						
. E – SCSI Enclosure Services device (SES)						
. B – Simplified Direct-Access (Reduced Block) device (RBC)						
. K – Optical Card Reader/Writer device (OCRW)						
. V – Automation/Device Interface device (ADC)						
. F – Object-based Storage Device (OSD)						
ASC	ASCQ	DTLPWROMAEBKVF	Description			
04h	11h	DT WRO AEB VF	LOGICAL UNIT NOT READY, NOTIFY (ENABLE SPINUP) REQUIRED			
04h	12h	M V	LOGICAL UNIT NOT READY, OFFLINE			
04h	07h	DTLPWROMAEBKVF	LOGICAL UNIT NOT READY, OPERATION IN PROGRESS			
04h	05h	DT W O A BK F	LOGICAL UNIT NOT READY, REBUILD IN PROGRESS			
04h	06h	DT W O A BK	LOGICAL UNIT NOT READY, RECALCULATION IN PROGRESS			
04h	13h	DT R MAEBKV	LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS			
04h	09h	DTLPWROMAEBKVF	LOGICAL UNIT NOT READY, SELF-TEST IN PROGRESS			
04h	0Dh	F	LOGICAL UNIT NOT READY, STRUCTURE CHECK REQUIRED			
25h	00h	DTLPWROMAEBKVF	LOGICAL UNIT NOT SUPPORTED			
27h	02h	DT WRO BK	LOGICAL UNIT SOFTWARE WRITE PROTECTED			
3Eh	04h	DTLPWROMAEBKVF	LOGICAL UNIT UNABLE TO UPDATE SELF-TEST LOG			
5Eh	00h	DTLPWRO A K	LOW POWER CONDITION ON			
55h	08h	T	MAXIMUM NUMBER OF SUPPLEMENTAL DECRYPTION KEYS EXCEEDED			
15h	01h	DTL WROM BK	MECHANICAL POSITIONING ERROR			
3Bh	16h	R	MECHANICAL POSITIONING OR CHANGER ERROR			
5Dh	01h	R B	MEDIA FAILURE PREDICTION THRESHOLD EXCEEDED			
53h	00h	DTL WROM BK	MEDIA LOAD OR EJECT FAILED			
6Fh	04h	R	MEDIA REGION CODE IS MISMATCHED TO LOGICAL UNIT REGION			
3Fh	11h	DT WROM B	MEDIUM AUXILIARY MEMORY ACCESSIBLE			
55h	09h	M	MEDIUM AUXILIARY MEMORY NOT ACCESSIBLE			
3Bh	0Dh	DT WROM BK	MEDIUM DESTINATION ELEMENT FULL			
31h	00h	DT WRO BK	MEDIUM FORMAT CORRUPTED			
3Fh	10h	DT WROM B	MEDIUM LOADABLE			
3Bh	13h	DT WROM BK	MEDIUM MAGAZINE INSERTED			
3Bh	14h	DT WROM BK	MEDIUM MAGAZINE LOCKED			
3Bh	11h	DT WROM BK	MEDIUM MAGAZINE NOT ACCESSIBLE			
3Bh	12h	DT WROM BK	MEDIUM MAGAZINE REMOVED			
3Bh	15h	DT WROM BK	MEDIUM MAGAZINE UNLOCKED			
30h	10h	R	MEDIUM NOT FORMATTED			
3Ah	00h	DTL WROM BK	MEDIUM NOT PRESENT			
3Ah	03h	DT WROM B	MEDIUM NOT PRESENT - LOADABLE			
3Ah	04h	DT WRO B	MEDIUM NOT PRESENT - MEDIUM AUXILIARY MEMORY ACCESSIBLE			
3Ah	01h	DT WROM BK	MEDIUM NOT PRESENT - TRAY CLOSED			
3Ah	02h	DT WROM BK	MEDIUM NOT PRESENT - TRAY OPEN			
53h	02h	DT WROM BK	MEDIUM REMOVAL PREVENTED			
53h	03h	M	MEDIUM REMOVAL PREVENTED BY DATA TRANSFER ELEMENT			
3Bh	0Eh	DT WROM BK	MEDIUM SOURCE ELEMENT EMPTY			
53h	04h	T	MEDIUM THREAD OR UNTHREAD FAILURE			
43h	00h	DTLPWROMAEBKVF	MESSAGE ERROR			
3Fh	01h	DTLPWROMAEBKVF	MICROCODE HAS BEEN CHANGED			
Annex D contains the ASC and ASCQ assignments in numeric order.						

**Table 41 — ASC and ASCQ assignments** (part 10 of 16)

D – Direct Access Block Device (SBC-3)							Device Column key
. T – Sequential Access Device (SSC-3)							blank = code not used
. L – Printer Device (SSC)							not blank = code used
. P – Processor Device (SPC-2)							
. . W – Write Once Block Device (SBC)							
. . R – C/DVD Device (MMC-6)							
. . O – Optical Memory Block Device (SBC)							
. . M – Media Changer Device (SMC-3)							
. . A – Storage Array Device (SCC-2)							
. . E – SCSI Enclosure Services device (SES)							
. . B – Simplified Direct-Access (Reduced Block) device (RBC)							
. . K – Optical Card Reader/Writer device (OCRW)							
. . V – Automation/Device Interface device (ADC)							
. . F – Object-based Storage Device (OSD)							
ASC	ASCQ	DTLPWROMAEBKVF	Description				
1Dh	00h	DT WRO BK	MISCOMPARE DURING VERIFY OPERATION				
11h	0Ah	DT O BK	MISCORRECTED ERROR				
2Ah	01h	DTL WROMAEBKVF	MODE PARAMETERS CHANGED				
67h	03h	A	MODIFICATION OF LOGICAL UNIT FAILED				
69h	01h	A	MULTIPLE LOGICAL UNIT FAILURES				
07h	00h	DTL WROM BK	MULTIPLE PERIPHERAL DEVICES SELECTED				
11h	03h	DT W O BK	MULTIPLE READ ERRORS				
67h	09h	A	MULTIPLY ASSIGNED LOGICAL UNIT				
4Bh	04h	DT PWROMAEBK	NAK RECEIVED				
00h	00h	DTLPWROMAEBKVF	NO ADDITIONAL SENSE INFORMATION				
00h	15h	R	NO CURRENT AUDIO STATUS TO RETURN				
32h	00h	D W O BK	NO DEFECT SPARE LOCATION AVAILABLE				
11h	09h	T	NO GAP FOUND				
01h	00h	D W O BK	NO INDEX/SECTOR SIGNAL				
72h	07h	R	NO MORE TEST ZONE EXTENSIONS ARE ALLOWED				
72h	05h	R	NO MORE TRACK RESERVATIONS ALLOWED				
06h	00h	D WROM BK	NO REFERENCE POSITION FOUND				
02h	00h	D WRO BK	NO SEEK COMPLETE				
03h	01h	T	NO WRITE CURRENT				
24h	06h	F	NONCE NOT UNIQUE				
24h	07h	F	NONCE TIMESTAMP OUT OF RANGE				
28h	00h	DTLPWROMAEBKVF	NOT READY TO READY CHANGE, MEDIUM MAY HAVE CHANGED				
2Ch	0Bh	T	NOT RESERVED				
00h	16h	DTLPWROMAEBKVF	OPERATION IN PROGRESS				
5Ah	01h	DT WROM BK	OPERATOR MEDIUM REMOVAL REQUEST				
5Ah	00h	DTLPWRO BK	OPERATOR REQUEST OR STATE CHANGE INPUT				
5Ah	03h	DT WRO A BK	OPERATOR SELECTED WRITE PERMIT				
5Ah	02h	DT WRO A BK	OPERATOR SELECTED WRITE PROTECT				
61h	02h		OUT OF FOCUS				
4Eh	00h	DTLPWROMAEBKVF	OVERLAPPED COMMANDS ATTEMPTED				
2Dh	00h	T	OVERWRITE ERROR ON UPDATE IN PLACE				
20h	05h	T	Obsolete				
24h	02h	T	Obsolete				
24h	03h	T	Obsolete				
63h	01h	R	PACKET DOES NOT FIT IN AVAILABLE SPACE				
3Bh	05h	L	PAPER JAM				
1Ah	00h	DTLPWROMAEBKVF	PARAMETER LIST LENGTH ERROR				
26h	01h	DTLPWROMAEBKVF	PARAMETER NOT SUPPORTED				
26h	02h	DTLPWROMAEBKVF	PARAMETER VALUE INVALID				
2Ah	00h	DTL WROMAEBKVF	PARAMETERS CHANGED				
Annex D contains the ASC and ASCQ assignments in numeric order.							

Table 41 — ASC and ASCQ assignments (part 11 of 16)

D – Direct Access Block Device (SBC-3) . T – Sequential Access Device (SSC-3) . L – Printer Device (SSC) . P – Processor Device (SPC-2) . . W – Write Once Block Device (SBC) . . R – C/DVD Device (MMC-6) . . O – Optical Memory Block Device (SBC) . . . M – Media Changer Device (SMC-3) . . . A – Storage Array Device (SCC-2) . . . E – SCSI Enclosure Services device (SES) . . . B – Simplified Direct-Access (Reduced Block) device (RBC) . . . K – Optical Card Reader/Writer device (OCRW) . . . V – Automation/Device Interface device (ADC) . . . F – Object-based Storage Device (OSD)										Device Column key blank = code not used not blank = code used
ASC	ASCQ	DTLPWROMAEBKVF	Description							
69h	02h				A					PARITY/DATA MISMATCH
1Fh	00h	D			O			K		PARTIAL DEFECT LIST TRANSFER
2Ch	0Ah								F	PARTITION OR COLLECTION CONTAINS USER OBJECTS
03h	00h	DTL	W		O			BK		PERIPHERAL DEVICE WRITE FAULT
27h	05h	T			R					PERMANENT WRITE PROTECT
2Ch	06h				R					PERSISTENT PREVENT CONFLICT
27h	04h	T			R					PERSISTENT WRITE PROTECT
47h	06h	DT				MAEBKVF				PHY TEST FUNCTION IN PROGRESS
50h	02h				T					POSITION ERROR RELATED TO TIMING
3Bh	0Ch	T								POSITION PAST BEGINNING OF MEDIUM
3Bh	0Bh									POSITION PAST END OF MEDIUM
15h	02h	DT	WRO					BK		POSITIONING ERROR DETECTED BY READ OF MEDIUM
73h	01h				R					POWER CALIBRATION AREA ALMOST FULL
73h	03h				R					POWER CALIBRATION AREA ERROR
73h	02h				R					POWER CALIBRATION AREA IS FULL
29h	01h	DTLPWROMAEBKVF								POWER ON OCCURRED
29h	00h	DTLPWROMAEBKVF								POWER ON, RESET, OR BUS DEVICE RESET OCCURRED
5Eh	41h					B				POWER STATE CHANGE TO ACTIVE
5Eh	47h					BK				POWER STATE CHANGE TO DEVICE CONTROL
5Eh	42h					B				POWER STATE CHANGE TO IDLE
5Eh	45h					B				POWER STATE CHANGE TO SLEEP
5Eh	43h					B				POWER STATE CHANGE TO STANDBY
42h	00h	D								POWER-ON OR SELF-TEST FAILURE (SHOULD USE 40 NN)
2Ch	07h	DTLPWROMAEBKVF								PREVIOUS BUSY STATUS
2Ch	09h	DTLPWROM				EBKVF				PREVIOUS RESERVATION CONFLICT STATUS
2Ch	08h	DTLPWROMAEBKVF								PREVIOUS TASK SET FULL STATUS
1Ch	01h	D			O			BK		PRIMARY DEFECT LIST NOT FOUND
2Ah	08h	DT	WROMAEBKVF							PRIORITY CHANGED
73h	05h				R					PROGRAM MEMORY AREA IS FULL
73h	04h				R					PROGRAM MEMORY AREA UPDATE FAILURE
00h	07h				T					PROGRAMMABLE EARLY WARNING DETECTED
47h	05h	DTLPWROMAEBKVF								PROTOCOL SERVICE CRC ERROR
55h	07h							F		QUOTA ERROR
40h	00h	D								RAM FAILURE (SHOULD USE 40 NN)
15h	00h	DTL	WROM					BK		RANDOM POSITIONING ERROR
73h	17h				R					RDZ IS FULL
11h	13h	DTLPWRO				AEBKVF				READ ERROR - FAILED RETRANSMISSION REQUEST
11h	14h	D								READ ERROR - LBA MARKED BAD BY APPLICATION CLIENT
11h	11h				R					READ ERROR - LOSS OF STREAMING
6Fh	03h				R					READ OF SCRAMBLED SECTOR WITHOUT AUTHENTICATION

Annex D contains the ASC and ASCQ assignments in numeric order.

Table 41 — ASC and ASCQ assignments (part 12 of 16)

D – Direct Access Block Device (SBC-3)														<u>Device Column key</u> blank = code not used not blank = code used	
. T – Sequential Access Device (SSC-3)															
. L – Printer Device (SSC)															
. P – Processor Device (SPC-2)															
. . W – Write Once Block Device (SBC)															
. . R – C/DVD Device (MMC-6)															
. . O – Optical Memory Block Device (SBC)															
. . . M – Media Changer Device (SMC-3)															
. . . A – Storage Array Device (SCC-2)															
. . . E – SCSI Enclosure Services device (SES)															
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)															
. . . . K – Optical Card Reader/Writer device (OCRW)															
. . . . V – Automation/Device Interface device (ADC)															
. . . . F – Object-based Storage Device (OSD)															
. . . . .															
. . . . .															
. . . . .															
ASC	ASCQ	DT	L	P	W	R	O	M	A	E	B	K	V	F	Description
3Bh	0Ah														READ PAST BEGINNING OF MEDIUM
3Bh	09h														READ PAST END OF MEDIUM
3Bh	17h												F		READ PAST END OF USER OBJECT
11h	01h	DT			W	R	O					BK			READ RETRIES EXHAUSTED
6Ch	00h									A					REBUILD FAILURE OCCURRED
6Dh	00h									A					RECALCULATE FAILURE OCCURRED
14h	01h	DT			W	R	O					BK			RECORD NOT FOUND
14h	06h	DT			W		O					BK			RECORD NOT FOUND - DATA AUTO-REALLOCATED
14h	05h	DT			W		O					BK			RECORD NOT FOUND - RECOMMEND REASSIGNMENT
14h	00h	DT	L		W	R	O					BK			RECORDED ENTITY NOT FOUND
18h	02h	D			W	R	O					BK			RECOVERED DATA - DATA AUTO-REALLOCATED
18h	05h	D			W	R	O					BK			RECOVERED DATA - RECOMMEND REASSIGNMENT
18h	06h	D			W	R	O					BK			RECOVERED DATA - RECOMMEND REWRITE
17h	05h	D			W	R	O					BK			RECOVERED DATA USING PREVIOUS SECTOR ID
18h	03h						R								RECOVERED DATA WITH CIRC
18h	07h	D			W		O					BK			RECOVERED DATA WITH ECC - DATA REWRITTEN
18h	01h	D			W	R	O					BK			RECOVERED DATA WITH ERROR CORR. & RETRIES APPLIED
18h	00h	DT			W	R	O					BK			RECOVERED DATA WITH ERROR CORRECTION APPLIED
18h	04h						R								RECOVERED DATA WITH L-EC
18h	08h						R								RECOVERED DATA WITH LINKING
17h	03h	DT			W	R	O					BK			RECOVERED DATA WITH NEGATIVE HEAD OFFSET
17h	00h	DT			W	R	O					BK			RECOVERED DATA WITH NO ERROR CORRECTION APPLIED
17h	02h	DT			W	R	O					BK			RECOVERED DATA WITH POSITIVE HEAD OFFSET
17h	01h	DT			W	R	O					BK			RECOVERED DATA WITH RETRIES
17h	04h				W	R	O					B			RECOVERED DATA WITH RETRIES AND/OR CIRC APPLIED
17h	06h	D			W		O					BK			RECOVERED DATA WITHOUT ECC - DATA AUTO-REALLOCATED
17h	09h	D			W	R	O					BK			RECOVERED DATA WITHOUT ECC - DATA REWRITTEN
17h	07h	D			W	R	O					BK			RECOVERED DATA WITHOUT ECC - RECOMMEND REASSIGNMENT
17h	08h	D			W	R	O					BK			RECOVERED DATA WITHOUT ECC - RECOMMEND REWRITE
1Eh	00h	D			W		O					BK			RECOVERED ID WITH ECC CORRECTION
3Fh	06h	DT			W	R	O	M	A	E	B				REDUNDANCY GROUP CREATED OR MODIFIED
3Fh	07h	DT			W	R	O	M	A	E	B				REDUNDANCY GROUP DELETED
6Bh	01h									A					REDUNDANCY LEVEL GOT BETTER
6Bh	02h									A					REDUNDANCY LEVEL GOT WORSE
2Ah	05h	DT	L	P	W	R	O	M	A	E					REGISTRATIONS PREEMPTED
67h	05h									A					REMOVE OF LOGICAL UNIT FAILED
3Fh	0Eh	DT	L	P	W	R	O	M	A	E					REPORTED LUNS DATA HAS CHANGED
3Bh	08h						T								REPOSITION ERROR
2Ah	03h	DT	L	P	W	R	O	M	A	E		K			RESERVATIONS PREEMPTED
2Ah	04h	DT	L	P	W	R	O	M	A	E					RESERVATIONS RELEASED
Annex D contains the ASC and ASCQ assignments in numeric order.															

Annex D contains the ASC and ASCQ assignments in numeric order.

Table 41 — ASC and ASCQ assignments (part 13 of 16)

D – Direct Access Block Device (SBC-3)										<u>Device Column key</u> blank = code not used not blank = code used
. T – Sequential Access Device (SSC-3)										
. L – Printer Device (SSC)										
. P – Processor Device (SPC-2)										
. W – Write Once Block Device (SBC)										
. R – C/DVD Device (MMC-6)										
. O – Optical Memory Block Device (SBC)										
. M – Media Changer Device (SMC-3)										
. A – Storage Array Device (SCC-2)										
. E – SCSI Enclosure Services device (SES)										
. B – Simplified Direct-Access (Reduced Block) device (RBC)										
. K – Optical Card Reader/Writer device (OCRW)										
. V – Automation/Device Interface device (ADC)										
. F – Object-based Storage Device (OSD)										
ASC	ASCQ	DTLPWROMAEBKVF	Description							
00h	1Ah	T	REWIND OPERATION IN PROGRESS							
36h	00h	L	RIBBON, INK, OR TONER FAILURE							
73h	06h		R	RMA/PMA IS ALMOST FULL						
72h	06h		R	RMZ EXTENSION IS NOT ALLOWED						
37h	00h	DTL	WROMAEBKVF	ROUNDED PARAMETER						
5Ch	00h	D	O	RPL STATUS CHANGE						
2Ah	14h	DT	R	MAEBKV	SA CREATION CAPABILITIES DATA HAS CHANGED					
74h	30h	DT	R	MAEBKV	SA CREATION PARAMETER NOT SUPPORTED					
74h	10h	DT	R	MAEBKV	SA CREATION PARAMETER VALUE INVALID					
74h	11h	DT	R	MAEBKV	SA CREATION PARAMETER VALUE REJECTED					
39h	00h	DTL	WROMAE	K	SAVING PARAMETERS NOT SUPPORTED					
62h	00h				SCAN HEAD POSITIONING ERROR					
29h	02h	DTLPWROMAEBKVF	SCSI BUS RESET OCCURRED							
47h	00h	DTLPWROMAEBKVF	SCSI PARITY ERROR							
47h	02h	DTLPWROMAEBKVF	SCSI PARITY ERROR DETECTED DURING ST DATA PHASE							
54h	00h		P	SCSI TO HOST SYSTEM INTERFACE FAILURE						
24h	04h			F	SECURITY AUDIT VALUE FROZEN					
74h	79h	D	SECURITY CONFLICT IN TRANSLATED DEVICE							
74h	00h	T	SECURITY ERROR							
24h	05h			F	SECURITY WORKING KEY FROZEN					
45h	00h	DTLPWROMAEBKVF	SELECT OR RESELECT FAILURE							
3Bh	00h	TL	SEQUENTIAL POSITIONING ERROR							
5Dh	45h	D		B	SERVO IMPENDING FAILURE ACCESS TIMES TOO HIGH					
5Dh	47h	D		B	SERVO IMPENDING FAILURE CHANNEL PARAMETRICS					
5Dh	48h	D		B	SERVO IMPENDING FAILURE CONTROLLER DETECTED					
5Dh	42h	D		B	SERVO IMPENDING FAILURE DATA ERROR RATE TOO HIGH					
5Dh	4Ch	D		B	SERVO IMPENDING FAILURE DRIVE CALIBRATION RETRY COUNT					
5Dh	41h	D		B	SERVO IMPENDING FAILURE DRIVE ERROR RATE TOO HIGH					
5Dh	40h	D		B	SERVO IMPENDING FAILURE GENERAL HARD DRIVE FAILURE					
5Dh	43h	D		B	SERVO IMPENDING FAILURE SEEK ERROR RATE TOO HIGH					
5Dh	4Ah	D		B	SERVO IMPENDING FAILURE SEEK TIME PERFORMANCE					
5Dh	4Bh	D		B	SERVO IMPENDING FAILURE SPIN-UP RETRY COUNT					
5Dh	46h	D		B	SERVO IMPENDING FAILURE START UNIT TIMES TOO HIGH					
5Dh	49h	D		B	SERVO IMPENDING FAILURE THROUGHPUT PERFORMANCE					
5Dh	44h	D		B	SERVO IMPENDING FAILURE TOO MANY BLOCK REASSIGNS					
72h	00h			R	SESSION FIXATION ERROR					
72h	03h			R	SESSION FIXATION ERROR - INCOMPLETE TRACK IN SESSION					
72h	01h			R	SESSION FIXATION ERROR WRITING LEAD-IN					
72h	02h			R	SESSION FIXATION ERROR WRITING LEAD-OUT					
00h	1Bh	T	SET CAPACITY OPERATION IN PROGRESS							

Annex D contains the ASC and ASCQ assignments in numeric order.

Annex D contains the ASC and ASCQ assignments in numeric order.

Table 41 — ASC and ASCQ assignments (part 14 of 16)

D – Direct Access Block Device (SBC-3)										Device Column key
. T – Sequential Access Device (SSC-3)										blank = code not used
. L – Printer Device (SSC)										not blank = code used
. P – Processor Device (SPC-2)										
. W – Write Once Block Device (SBC)										
. R – C/DVD Device (MMC-6)										
. O – Optical Memory Block Device (SBC)										
. M – Media Changer Device (SMC-3)										
. A – Storage Array Device (SCC-2)										
. E – SCSI Enclosure Services device (SES)										
. B – Simplified Direct-Access (Reduced Block) device (RBC)										
. K – Optical Card Reader/Writer device (OCRW)										
. V – Automation/Device Interface device (ADC)										
. F – Object-based Storage Device (OSD)										
ASC	ASCQ	DTLPWROMAEBKVF	Description							
67h	0Ah	DTLPWROMAEBKVF	SET TARGET PORT GROUPS COMMAND FAILED							
00h	03h	T	SETMARK DETECTED							
3Bh	04h	L	SLEW FAILURE							
47h	7Fh	DT PWROMAEBK	SOME COMMANDS CLEARED BY ISCSI PROTOCOL EVENT							
5Dh	03h	R	SPARE AREA EXHAUSTION PREDICTION THRESHOLD EXCEEDED							
3Fh	08h	DT WROMAEB	SPARE CREATED OR MODIFIED							
3Fh	09h	DT WROMAEB	SPARE DELETED							
5Dh	55h	D	B	SPINDLE IMPENDING FAILURE ACCESS TIMES TOO HIGH						
5Dh	57h	D	B	SPINDLE IMPENDING FAILURE CHANNEL PARAMETRICS						
5Dh	58h	D	B	SPINDLE IMPENDING FAILURE CONTROLLER DETECTED						
5Dh	52h	D	B	SPINDLE IMPENDING FAILURE DATA ERROR RATE TOO HIGH						
5Dh	5Ch	D	B	SPINDLE IMPENDING FAILURE DRIVE CALIBRATION RETRY COUNT						
5Dh	51h	D	B	SPINDLE IMPENDING FAILURE DRIVE ERROR RATE TOO HIGH						
5Dh	50h	D	B	SPINDLE IMPENDING FAILURE GENERAL HARD DRIVE FAILURE						
5Dh	53h	D	B	SPINDLE IMPENDING FAILURE SEEK ERROR RATE TOO HIGH						
5Dh	5Ah	D	B	SPINDLE IMPENDING FAILURE SEEK TIME PERFORMANCE						
5Dh	5Bh	D	B	SPINDLE IMPENDING FAILURE SPIN-UP RETRY COUNT						
5Dh	56h	D	B	SPINDLE IMPENDING FAILURE START UNIT TIMES TOO HIGH						
5Dh	59h	D	B	SPINDLE IMPENDING FAILURE THROUGHPUT PERFORMANCE						
5Dh	54h	D	B	SPINDLE IMPENDING FAILURE TOO MANY BLOCK REASSIGNS						
09h	03h	WRO	SPINDLE SERVO FAILURE							
5Ch	02h	D O	SPINDLES NOT SYNCHRONIZED							
5Ch	01h	D O	SPINDLES SYNCHRONIZED							
5Eh	04h	DTLPWRO A K	STANDBY CONDITION ACTIVATED BY COMMAND							
5Eh	02h	DTLPWRO A K	STANDBY CONDITION ACTIVATED BY TIMER							
6Bh	00h	A	STATE CHANGE HAS OCCURRED							
1Bh	00h	DTLPWROMAEBKVF	SYNCHRONOUS DATA TRANSFER ERROR							
55h	01h	D O BK	SYSTEM BUFFER FULL							
55h	00h	P	SYSTEM RESOURCE FAILURE							
4Dh	NNh	DTLPWROMAEBKVF	TAGGED OVERLAPPED COMMANDS (NN = TASK TAG)							
33h	00h	T	TAPE LENGTH ERROR							
3Bh	03h	L	TAPE OR ELECTRONIC VERTICAL FORMS UNIT NOT READY							
3Bh	01h	T	TAPE POSITION ERROR AT BEGINNING-OF-MEDIUM							
3Bh	02h	T	TAPE POSITION ERROR AT END-OF-MEDIUM							
3Fh	00h	DTLPWROMAEBKVF	TARGET OPERATING CONDITIONS HAVE CHANGED							
0Dh	01h	DTLPWRO A K	THIRD PARTY DEVICE FAILURE							
5Bh	01h	DTLPWROM K	THRESHOLD CONDITION MET							
26h	03h	DTLPWROMAE K	THRESHOLD PARAMETERS NOT SUPPORTED							
3Eh	02h	DTLPWROMAEBKVF	TIMEOUT ON LOGICAL UNIT							
2Ah	10h	DT M E V	TIMESTAMP CHANGED							
Annex D contains the ASC and ASCQ assignments in numeric order.										

Annex D contains the ASC and ASCQ assignments in numeric order.



**Table 41 — ASC and ASCQ assignments** (part 15 of 16)

D – Direct Access Block Device (SBC-3)				Device Column key
. T – Sequential Access Device (SSC-3)				blank = code not used
. L – Printer Device (SSC)				not blank = code used
. P – Processor Device (SPC-2)				
. W – Write Once Block Device (SBC)				
. R – C/DVD Device (MMC-6)				
. O – Optical Memory Block Device (SBC)				
. M – Media Changer Device (SMC-3)				
. A – Storage Array Device (SCC-2)				
. E – SCSI Enclosure Services device (SES)				
. B – Simplified Direct-Access (Reduced Block) device (RBC)				
. K – Optical Card Reader/Writer device (OCRW)				
. V – Automation/Device Interface device (ADC)				
. F – Object-based Storage Device (OSD)				
ASC	ASCQ	DTLPWROMAEBKVF	Description	
26h	08h	DTLPWRO	K	TOO MANY SEGMENT DESCRIPTORS
26h	06h	DTLPWRO	K	TOO MANY TARGET DESCRIPTORS
2Ch	01h			TOO MANY WINDOWS SPECIFIED
4Bh	02h	DT PWROMAEBK		TOO MUCH WRITE DATA
09h	00h	DT WRO	B	TRACK FOLLOWING ERROR
09h	01h	WRO	K	TRACKING SERVO FAILURE
29h	06h	DTLPWROMAEBKVF		TRANSCIEVER MODE CHANGED TO LVD
29h	05h	DTLPWROMAEBKVF		TRANSCIEVER MODE CHANGED TO SINGLE-ENDED
61h	01h			UNABLE TO ACQUIRE VIDEO
74h	01h	T		UNABLE TO DECRYPT DATA
74h	0Ch	DT R MAEBKV		UNABLE TO DECRYPT PARAMETER LIST
57h	00h	R		UNABLE TO RECOVER TABLE-OF-CONTENTS
74h	02h	T		UNENCRYPTED DATA ENCOUNTERED WHILE DECRYPTING
26h	0Ah	DTLPWRO	K	UNEXPECTED INEXACT SEGMENT
74h	06h	T		UNKNOWN SIGNATURE VERIFICATION KEY
53h	01h	T		UNLOAD TAPE FAILURE
08h	04h	DTLPWRO	K	UNREACHABLE COPY TARGET
11h	00h	DT WRO	BK	UNRECOVERED READ ERROR
11h	04h	D W O	BK	UNRECOVERED READ ERROR - AUTO REALLOCATE FAILED
11h	0Bh	D W O	BK	UNRECOVERED READ ERROR - RECOMMEND REASSIGNMENT
11h	0Ch	D W O	BK	UNRECOVERED READ ERROR - RECOMMEND REWRITE THE DATA
46h	00h	DTLPWROM	BK	UNSUCCESSFUL SOFT RESET
35h	01h	DTLPWROMAEBKVF		UNSUPPORTED ENCLOSURE FUNCTION
26h	09h	DTLPWRO	K	UNSUPPORTED SEGMENT DESCRIPTOR TYPE CODE
26h	07h	DTLPWRO	K	UNSUPPORTED TARGET DESCRIPTOR TYPE CODE
59h	00h	O		UPDATED BLOCK READ
26h	12h	T		VENDOR SPECIFIC KEY REFERENCE NOT FOUND
00h	1Ch	T		VERIFY OPERATION IN PROGRESS
61h	00h			VIDEO ACQUISITION ERROR
65h	00h	DTLPWROMAEBKVF		VOLTAGE FAULT
3Fh	0Ah	DT WROMAEBK		VOLUME SET CREATED OR MODIFIED
3Fh	0Ch	DT WROMAEBK		VOLUME SET DEASSIGNED
3Fh	0Bh	DT WROMAEBK		VOLUME SET DELETED
3Fh	0Dh	DT WROMAEBK		VOLUME SET REASSIGNED
0Bh	00h	DTLPWROMAEBKVF		WARNING
0Bh	05h	DTLPWRO AEBKVF		WARNING - BACKGROUND MEDIUM SCAN DETECTED MEDIUM ERROR
0Bh	04h	DTLPWRO AEBKVF		WARNING - BACKGROUND PRE-SCAN DETECTED MEDIUM ERROR
0Bh	03h	DTLPWROMAEBKVF		WARNING - BACKGROUND SELF-TEST FAILED
0Bh	07h	DTLPWROMAEBKVF		WARNING - DEGRADED POWER TO NON-VOLATILE CACHE
0Bh	02h	DTLPWROMAEBKVF		WARNING - ENCLOSURE DEGRADED
Annex D contains the ASC and ASCQ assignments in numeric order.				

**Table 41 — ASC and ASCQ assignments** (part 16 of 16)

D – Direct Access Block Device (SBC-3)										Device Column key						
. T – Sequential Access Device (SSC-3)										blank = code not used						
. L – Printer Device (SSC)										not blank = code used						
. P – Processor Device (SPC-2)																
. . W – Write Once Block Device (SBC)																
. . R – C/DVD Device (MMC-6)																
. . O – Optical Memory Block Device (SBC)																
. . . M – Media Changer Device (SMC-3)																
. . . A – Storage Array Device (SCC-2)																
. . . E – SCSI Enclosure Services device (SES)																
. . . B – Simplified Direct-Access (Reduced Block) device (RBC)																
. . . K – Optical Card Reader/Writer device (OCRW)																
. . . V – Automation/Device Interface device (ADC)																
. . . F – Object-based Storage Device (OSD)																
. . . . .																
ASC	ASCQ	DTLPWROMAEBKVF	Description													
0Bh	06h	DTLPWROMAEBKVF	WARNING - NON-VOLATILE CACHE NOW VOLATILE													
0Bh	01h	DTLPWROMAEBKVF	WARNING - SPECIFIED TEMPERATURE EXCEEDED													
30h	0Dh	T	WORM MEDIUM - INTEGRITY CHECK													
30h	0Ch	T	WORM MEDIUM - OVERWRITE ATTEMPTED													
50h	00h	T	WRITE APPEND ERROR													
50h	01h	T	WRITE APPEND POSITION ERROR													
0Ch	00h	T R	WRITE ERROR													
0Ch	02h	D W O BK	WRITE ERROR - AUTO REALLOCATION FAILED													
0Ch	09h	R	WRITE ERROR - LOSS OF STREAMING													
0Ch	0Dh	DTLPWRO AEBKVF	WRITE ERROR - NOT ENOUGH UNSOLICITED DATA													
0Ch	0Ah	R	WRITE ERROR - PADDING BLOCKS ADDED													
0Ch	03h	D W O BK	WRITE ERROR - RECOMMEND REASSIGNMENT													
0Ch	01h	K	WRITE ERROR - RECOVERED WITH AUTO REALLOCATION													
0Ch	08h	R	WRITE ERROR - RECOVERY FAILED													
0Ch	07h	R	WRITE ERROR - RECOVERY NEEDED													
0Ch	0Ch	DTLPWRO AEBKVF	WRITE ERROR - UNEXPECTED UNSOLICITED DATA													
27h	00h	DT WRO BK	WRITE PROTECTED													
31h	02h	R	ZONED FORMATTING FAILED DUE TO SPARE LINKING													
3Fh	12h	DTLPWR MAEBK F	iSCSI IP ADDRESS ADDED													
3Fh	14h	DTLPWR MAEBK F	iSCSI IP ADDRESS CHANGED													
3Fh	13h	DTLPWR MAEBK F	iSCSI IP ADDRESS REMOVED													
80h	xxh	\	Vendor specific													
Through		>														
FFh	xxh	/	Vendor specific qualification of standard ASC													
xxh	80h	\														
Through		>														
xxh	FFh	/														
All codes not shown are reserved.																
Annex D contains the ASC and ASCQ assignments in numeric order.																

## 4.6 Secure random numbers

Secure Random numbers should be generated as specified by RFC 4086 (e.g., see FIPS 140-2 Annex C: Approved Random Number Generators).

If the same random number source is used generate random numbers for multiple purposes (e.g., nonces and secret keys), interactions between the two shall not be allowed to compromise secrecy. If the value sequence generated by the common random number source is predictable to any degree, then the random number values that are transmitted outside the SCSI device may provide information about the random number values that the SCSI device maintains internally, based on the reasonable assumption that an adversary knows the order in which the random numbers are obtained from the common random number source. SCSI devices shall eliminate sources of such predictability.

Compliance with RFC 4086 is one method for achieving the required independence between random number values.

## **5 Model common to all device types**

### **5.1 Introduction to the model common to all device types**

This model describes some of the general characteristics expected of most SCSI devices. It is not intended to alter any requirements defined elsewhere in SCSI. Devices conforming to this standard also shall conform to SAM-4.

### **5.2 Important commands for all SCSI device servers**

#### **5.2.1 Commands implemented by all SCSI device servers**

This standard defines three commands that all SCSI device servers shall implement - INQUIRY, REPORT LUNS, and TEST UNIT READY. These commands are used to discover a logical unit's capabilities, to discover the system configuration, and to determine whether a logical unit is ready.

#### **5.2.2 Commands recommended for all SCSI device servers**

Support for the REQUEST SENSE command is recommended to provide compatibility with application clients designed to use previous versions of this standard or status polling features defined by command standards (see 3.1.27).

#### **5.2.3 Using the INQUIRY command**

The INQUIRY command (see 6.4) may be used by an application client to determine the configuration of a logical unit. Device servers respond with information that includes their device type and standard version and may include the vendor's identification, model number and other information.

The Device Identification VPD page (see 7.7.3) returned in response to an INQUIRY command with the EVPD bit set to one and the PAGE CODE field set to 83h contains identifying information for the logical unit, the target port, and the SCSI target device.

It is recommended that device servers be capable of returning this information, or whatever part of it that is available, upon completing power-on initialization. A device server may take longer to get certain portions of this information, especially if it retrieves the information from the medium.

#### **5.2.4 Using the REPORT LUNS command**

The REPORT LUNS command (see 6.23) may be used by an application client to discover the logical unit inventory (see 3.1.81) that is accessible to the I\_T nexus on which the command is sent.

#### **5.2.5 Using the TEST UNIT READY command**

The TEST UNIT READY command (see 6.37) allows an application client to poll a logical unit until it is ready without the need to allocate space for returned data. The TEST UNIT READY command may be used to check the media status of logical units with removable media. Device servers should respond promptly to indicate the current status of the SCSI device.

NOTE 8 - Delays to achieve GOOD status from a TEST UNIT READY command may adversely affect initiator device performance.

### 5.2.6 Using the REQUEST SENSE command

The REQUEST SENSE command (see 6.29) may be used by an application client to poll the status of some background operations and to clear interlocked unit attention conditions (see 7.4.6).

### 5.3 Implicit head of queue

Each of the following commands may be processed by the task manager as if it has a task attribute of HEAD OF QUEUE (see SAM-4) if it is received with a SIMPLE task attribute, an ORDERED task attribute, or no task attribute:

- a) INQUIRY; and
- b) REPORT LUNS.

An application client should not send a command with the ORDERED task attribute if the command may be processed as if it has a task attribute of HEAD OF QUEUE because whether the ORDERED task attribute is honored for these commands is vendor specific.

### 5.4 Parameter rounding

Certain parameters sent to a device server with various commands contain a range of values. Device servers may choose to implement only selected values from this range. When the device server receives a value that it does not support, it either rejects the command (i.e., CHECK CONDITION status with ILLEGAL REQUEST sense key) or it rounds the value received to a supported value.

When parameter rounding is implemented, a device server that receives a parameter value that is not an exact supported value shall adjust the value to one that it supports and shall return CHECK CONDITION status, with the sense key set to RECOVERED ERROR, and the additional sense code set to ROUNDED PARAMETER. The application client should send an appropriate command to learn what value the device server has selected.

The device server shall reject unsupported values unless rounding is permitted in the description of the parameter. When the description of a parameter states that rounding is permitted, the device server should adjust maximum-value fields down to the next lower supported value than the one specified by the application client. Minimum-value fields should be rounded up to the next higher supported value than the one specified by the application client. In some cases, the type of rounding (i.e., up or down) is specified in the description of the parameter.

### 5.5 Parsing variable-length parameter lists and parameter data

Parameter lists and parameter data (e.g., diagnostic pages, mode pages, log pages, and VPD pages) often include length fields indicating the size of the parameter list or parameter data (e.g., the MODE DATA LENGTH field in the mode parameter header (see 7.4.3)). Parameter lists and parameter data often include descriptor lists and descriptor length fields containing the length of the descriptors in the descriptor lists (e.g., the DESIGNATOR LENGTH field in the designation descriptor used in the Device Identification VPD page (see 7.7.3.1)).

An application client or device server shall not assume that any length field contains the value defined in a SCSI standard.

If a device server receives a parameter list containing a length field (e.g., a PAGE LENGTH field) and containing more bytes than are defined in the standard to which it was designed (e.g., the device server complies with a version of a SCSI standard defining that a parameter list has 24 bytes, but receives a parameter list containing 36 bytes), then

the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

For parameter lists containing a descriptor length field and a descriptor list, if a device server receives more bytes in a descriptor than are defined in the standard to which it was designed (e.g., the device server complies with a version of a SCSI standard defining that a descriptor is 12 bytes, but receives a parameter list containing a 16 byte form of that descriptor), then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

An application client should ignore any bytes of parameter data beyond those defined in the standard to which it was designed (e.g., if the application client complies with a version of a SCSI standard defining 24 bytes of parameter data, but receives 36 bytes of parameter data, then the application client should ignore the last 12 bytes or the parameter data).

For additional response bytes containing a descriptor length field and a descriptor list, an application client should ignore any bytes in each descriptor beyond those defined in the standard to which it was designed (e.g., if the application client complies with a version of a SCSI standard defining that a descriptor has 24 bytes, but receives parameter data containing a descriptor list with a 36 byte form of that descriptor, then the application client should ignore the last 12 bytes of the descriptor).

## 5.6 Self-test operations

### 5.6.1 Default self-test

The SEND DIAGNOSTIC command provides a means to request that a SCSI device perform a self test. While the test is vendor specific, the means of requesting the test is standardized.

The default self-test is mandatory for all device types that support the SEND DIAGNOSTIC command. The response is GOOD status if the test detects no exceptions, or CHECK CONDITION status if the test detects exceptions.

### 5.6.2 The short and extended self-tests

There are two optional types of self-test aside from the mandatory default self-test that may be invoked using the SELF-TEST CODE field in the SEND DIAGNOSTIC command; a short self-test and an extended self-test. The goal of the short self-test is to quickly identify if the logical unit determines that it is faulty. A goal of the extended self-test routine is to simplify factory testing during integration by having logical units perform more comprehensive testing without application client intervention. A second goal of the extended self-test is to provide a more comprehensive test to validate the results of a short self-test, if its results are judged by the application client to be inconclusive.

The criteria for the short self-test are that it has one or more segments and completes in two minutes or less. The criteria for the extended self-test are that it has one or more segments and that the completion time is vendor specific. Any tests performed in the segments are vendor specific.

The following are examples of segments:

- a) An electrical segment wherein the logical unit tests its own electronics. The tests in this segment are vendor specific, but some examples of tests that may be included are:
  - A) A buffer RAM test;
  - B) A read/write circuitry test; and/or
  - C) A test of the read/write heads;

- b) A seek/servo segment wherein a device tests its capability to find and servo on data tracks; and
- c) A read/verify scan segment wherein a device performs read scanning of some or all of the medium surface.

The tests performed in the segments may be the same for the short and extended self-tests. The time required by a logical unit to complete its extended self-test is reported in the EXTENDED SELF-TEST COMPLETION TIME field in the Control mode page (see 7.4.6).

### 5.6.3 Self-test modes

#### 5.6.3.1 Self-test modes overview

Both a foreground mode (see 5.6.3.2) and a background mode (see 5.6.3.3) is defined for both short and extended self-tests.

#### 5.6.3.2 Foreground mode

When a device server receives a SEND DIAGNOSTIC command specifying a self-test to be performed in the foreground mode, the device server shall return status for that command after the self-test has been completed.

While performing a self-test in the foreground mode, the device server shall terminate all commands except INQUIRY, REPORT LUNS, and REQUEST SENSE with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, SELF-TEST IN PROGRESS. If the device server receives an INQUIRY command, a REPORT LUNS command, or a REQUEST SENSE command while performing a self-test in the foreground mode, then the device server shall process the command.

If a device server is performing a self-test in the foreground mode and a test segment error occurs during the test, the device server shall update the Self-Test Results log page (see 7.2.11) and terminate the SEND DIAGNOSTIC command with CHECK CONDITION status, with the sense key set to HARDWARE ERROR, and the additional sense code set to LOGICAL UNIT FAILED SELF-TEST. The application client may obtain additional information about the failure by reading the Self-Test Results log page. If the device server is unable to update the Self-Test Results log page, it shall terminate the SEND DIAGNOSTIC command with CHECK CONDITION status, with the sense key set to HARDWARE ERROR, and the additional sense code set to LOGICAL UNIT UNABLE TO UPDATE SELF-TEST LOG.

An application client should reserve the logical unit before initiating a self-test in the foreground mode. An application client may terminate a self-test that is being performed in the foreground mode using commands (see clause 6) or task management functions (see SAM-4) (e.g., a PERSISTENT RESERVE OUT command with PREEMPT AND ABORT service action, an ABORT TASK task management function, a CLEAR TASK SET task management function). In addition, a foreground mode self-test shall be terminated by an I\_T nexus loss (see SAM-4). If a SEND DIAGNOSTIC command that requested a self-test in the foreground mode is terminated while the SCSI target device is performing the self-test, the device server shall abort the self-test and update the Self-Test Results log page (see 7.2.11).

#### 5.6.3.3 Background mode

When a device server receives a SEND DIAGNOSTIC command specifying a self-test to be performed in the background mode, the device server shall return status for that command as soon as the CDB has been validated.

After returning status for the SEND DIAGNOSTIC command specifying a self-test to be performed in the background mode, the device server shall initialize the Self-Test Results log page (see 7.2.11) as follows. The self-test code from the SEND DIAGNOSTIC command shall be placed in the SELF-TEST CODE field in the log page. The SELF-TEST RESULTS field shall be set to Fh. After the Self-Test Results log page is initialized, the device server shall begin the first self-test segment.

While the device server is performing a self-test in the background mode, it shall terminate with CHECK CONDITION status any SEND DIAGNOSTIC command it receives that meets one of the following criteria:

- a) The SELFTEST bit is set to one; or
- b) The SELF-TEST CODE field contains a value other than 000b or 100b.

When terminating the SEND DIAGNOSTIC command, the sense key shall be set to NOT READY and the additional sense code shall be set to LOGICAL UNIT NOT READY, SELF-TEST IN PROGRESS.

While performing a self-test in the background mode, the device server shall suspend the self-test to service any other commands received with the exceptions listed in table 42. Suspension of the self-test to service the command shall occur as soon as practical and shall not take longer than two seconds.

**Table 42 — Exception commands for background self-tests**

Device type	Command	Reference
All device types	SEND DIAGNOSTIC (with SELF-TEST CODE field set to 100b) WRITE BUFFER (with the mode set to any download microcode option)	6.32 6.39
Direct access block	FORMAT UNIT START STOP UNIT	SBC-3
Sequential access	ERASE FORMAT MEDIUM LOAD UNLOAD LOCATE READ READ POSITION READ REVERSE  REWIND SPACE VERIFY WRITE WRITE BUFFER WRITE FILEMARKS	SSC-3
Media changer	EXCHANGE MEDIUM INITIALIZE ELEMENT STATUS MOVE MEDIUM POSITION TO ELEMENT READ ELEMENT STATUS (if CURDATA=0 and device motion is required) WRITE BUFFER	SMC-3
Object-based storage	Any command with operation code 7Fh (i.e., all commands defined by the OSD standard)	OSD
NOTE Device types not listed in this table do not have commands that are exceptions for background self-tests, other than those listed above for all device types.		

If one of the exception commands listed in table 42 is received, the device server shall abort the self-test, update the self-test log, and service the command as soon as practical but not longer than two seconds after the CDB has been validated.

An application client may terminate a self-test that is being performed in the background mode by issuing a SEND DIAGNOSTIC command with the SELF-TEST CODE field set to 100b (i.e., abort background self-test function). A background mode self-test shall not be terminated by an I\_T nexus loss (see SAM-4).

#### 5.6.3.4 Features common to foreground and background self-test modes

The PROGRESS INDICATION field in parameter data returned in response to a REQUEST SENSE command (see 6.29) may be used by the application client at any time during a self-test operation to poll the logical unit's progress. While a self-test operation is in progress unless an error has occurred, a device server shall respond to a



REQUEST SENSE command by returning parameter data containing sense data with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, SELF-TEST IN PROGRESS with the sense key specific bytes set for progress indication.

An application client may use the EBACKERR bit and the MRIE field in the Informational Exception Control mode page (see 7.4.11) to control the reporting of errors that occur during a background self-test operation.

The application client may obtain information about the 20 most recently completed self-tests by reading the Self-Test Results log page (see 7.2.11). This is the only method for an application client to obtain information about self-tests performed in the background mode unless an error occurs.

Table 43 summarizes when a logical unit returns status after receipt of a self-test command, how an application client may abort a self-test, how a logical unit handles commands that are entered into the task set while a self-test is in progress, and how a logical unit reports a self-test failure.

**Table 43 — Self-test mode summary**

<b>Mode</b>	<b>When Status is Returned</b>	<b>How to abort the self-test</b>	<b>Processing of subsequent commands while self-test is being processed</b>	<b>Self-test failure reporting</b>
Fore-ground	After the self-test is complete	One of the commands (see 5.6.3.2) and task management functions (see SAM-4) that cause tasks to be aborted	If the command is INQUIRY, REPORT LUNS or REQUEST SENSE, process normally. Otherwise, terminate with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, SELF-TEST IN PROGRESS.	Terminate with CHECK CONDITION status, with the sense key set to HARDWARE ERROR, and the additional sense code set to LOGICAL UNIT FAILED SELF-TEST or LOGICAL UNIT UNABLE TO UPDATE SELF-TEST LOG.
Back-ground	After the CDB is validated	SEND DIAGNOSTIC command with SELF-TEST CODE field set to 100b	Process the command, except as described in 5.6.3.3.	Application client: a) Checks Self-Test Results log page (see 7.2.11) after the PROGRESS INDICATION field returned from REQUEST SENSE indicates the self-test is complete; or b) Uses the EBACKERR bit and the MRIE field (see 7.4.11) to specify a method of indicating a failure occurred. If a failure occurs an additional sense code of WARNING - BACKGROUND SELF-TEST FAILED shall be returned using the method defined in the MRIE field. <sup>a</sup>
<sup>a</sup> The device server shall not report an error until after the Self-Test Results log page is updated.				

## 5.7 Reservations

### 5.7.1 Persistent Reservations overview

Reservations may be used to allow a device server to process commands from a selected set of I\_T nexuses (i.e., combinations of initiator ports accessing target ports) and reject commands from I\_T nexuses outside the selected set. The device server uniquely identifies I\_T nexuses using protocol specific mechanisms.

Application clients may add or remove I\_T nexuses from the selected set using reservation commands. If the application clients do not cooperate in the reservation protocol, data may be unexpectedly modified and deadlock conditions may occur.

The persistent reservations mechanism allows multiple application clients communicating through multiple I\_T nexuses to preserve reservation operations across SCSI initiator device failures, which usually involve logical unit resets and involve I\_T nexus losses. Persistent reservations persist across recovery actions. Persistent reservations are not reset by hard reset, logical unit reset, or I\_T nexus loss.

The persistent reservation held by a failing I\_T nexus may be preempted by another I\_T nexus as part of its recovery process. Persistent reservations shall be retained by the device server until released, preempted, or cleared by mechanisms specified in this standard. Optionally, persistent reservations may be retained when power to the SCSI target device is removed.

The PERSISTENT RESERVE OUT and PERSISTENT RESERVE IN commands provide the basic mechanism for dynamic contention resolution in systems with multiple initiator ports accessing a logical unit.

Before a persistent reservation may be established, the application client shall register a reservation key for each I\_T nexus with the device server. Reservation keys are necessary to allow:

- a) Authentication of subsequent PERSISTENT RESERVE OUT commands;
- b) Identification of other I\_T nexuses that are registered;
- c) Identification of the reservation key(s) that have an associated persistent reservation;
- d) Preemption of a persistent reservation from a failing or uncooperative I\_T nexus; and
- e) Multiple I\_T nexuses to participate in a persistent reservation.

The reservation key provides a method for the application client to associate a protocol-independent identifier with a registered I\_T nexus. The reservation key is used in the PERSISTENT RESERVE IN command to identify which I\_T nexuses are registered and which I\_T nexus, if any, holds the persistent reservation. The reservation key is used in the PERSISTENT RESERVE OUT command to register an I\_T nexus, to verify the I\_T nexus being used for the PERSISTENT RESERVE OUT command is registered, and to specify which registrations or persistent reservation to preempt.

Reservation key values may be used by application clients to identify registered I\_T nexuses, using application specific methods that are outside the scope of this standard. This standard provides the ability to register no more than one reservation key per I\_T nexus. Multiple initiator ports may use the same reservation key value for a logical unit accessed through the same target ports. An initiator port may use the same reservation key value for a logical unit accessed through different target ports. The logical unit shall maintain a separate reservation key for each I\_T nexus, regardless of the reservation key's value.

An application client may register an I\_T nexus with multiple logical units in a SCSI target device using any combination of unique or duplicate reservation keys. These rules provide the ability for an application client to preempt multiple I\_T nexuses with a single PERSISTENT RESERVE OUT command, but they do not provide the ability for the application client to uniquely identify the I\_T nexuses using the PERSISTENT RESERVE commands.

See table 170 in 6.14.2 for a list of PERSISTENT RESERVE OUT service actions. See table 158 in 6.13.1 for a list of PERSISTENT RESERVE IN service actions.

The scope (see 6.13.3.3) of a persistent reservation shall be the entire logical unit.

The type (see 6.13.3.4) of a persistent reservation defines the selected set of I\_T nexuses for which the persistent reservation places restrictions on commands.

The details of which commands are allowed under what types of reservations are described in table 44.

In table 44 and table 45 the following key words are used:

**allowed:** Commands received from I\_T nexuses not holding the reservation or from I\_T nexuses not registered when a registrants only or all registrants type persistent reservation is present should complete normally.

**conflict:** Commands received from I\_T nexuses not holding the reservation or from I\_T nexuses not registered when a registrants only or all registrants type persistent reservation is present shall not be performed and the device server shall complete the command with RESERVATION CONFLICT status.

Commands from I\_T nexuses holding a reservation should complete normally. The behavior of commands from registered I\_T nexuses when a registrants only or all registrants type persistent reservation is present is specified in table 44 and table 45.

A command shall be checked for reservation conflicts before the task containing that command enters the enabled task state. Once a task has entered the enabled task state, the command that comprises the task shall not be completed with RESERVATION CONFLICT status due to a subsequent reservation.

For each command, this standard or a command standard (see 3.1.27) defines the conditions that result in the command being completed with RESERVATION CONFLICT. Command standards define the conditions either in the device model or in the descriptions each of specific command.

**Table 44 — SPC commands that are allowed in the presence of various reservations (part 1 of 3)**

Command	Addressed logical unit has this type of persistent reservation held by another I_T nexus				
	From any I_T nexus		From registered I_T nexus (RR all types)	From not registered I_T nexus	
	Write Excl	Excl Access		Write Excl RR	Excl Access – RR
ACCESS CONTROL IN	Allowed	Allowed	Allowed	Allowed	Allowed
ACCESS CONTROL OUT	Allowed	Allowed	Allowed	Allowed	Allowed
CHANGE ALIASES	Conflict	Conflict	Allowed	Conflict	Conflict
Key: <b>Excl</b> =Exclusive, <b>RR</b> =Registrants Only or All Registrants, <> Not Equal					
<sup>a</sup> Exceptions to the behavior of the RESERVE and RELEASE commands described in SPC-2 are defined in 5.7.3. <sup>b</sup> Logical units claiming compliance with previous versions of this standard (e.g., SPC-2, SPC-3) may return RESERVATION CONFLICT in this case. Logical units may report whether certain commands are allowed in ALLOW COMMANDS field of the parameter data returned by the PERSISTENT RESERVE IN command with REPORT CAPABILITIES service action (see 6.13.4).					

Table 44 — SPC commands that are allowed in the presence of various reservations (part 2 of 3)

Command	Addressed logical unit has this type of persistent reservation held by another I_T nexus				
	From any I_T nexus		From registered I_T nexus (RR all types)	From not registered I_T nexus	
	Write Excl	Excl Access		Write Excl RR	Excl Access – RR
EXTENDED COPY	Conflict	Conflict	Allowed	Conflict	Conflict
INQUIRY	Allowed	Allowed	Allowed	Allowed	Allowed
LOG SELECT	Conflict	Conflict	Allowed	Conflict	Conflict
LOG SENSE	Allowed	Allowed	Allowed	Allowed	Allowed
MANAGEMENT PROTOCOL IN	Allowed	Conflict	Allowed	Allowed	Conflict
MANAGEMENT PROTOCOL OUT	Conflict	Conflict	Allowed	Conflict	Conflict
MODE SELECT(6) / MODE SELECT(10)	Conflict	Conflict	Allowed	Conflict	Conflict
MODE SENSE(6) / MODE SENSE(10)	Allowed <sup>b</sup>	Conflict	Allowed	Allowed <sup>b</sup>	Conflict
PERSISTENT RESERVE IN	Allowed	Allowed	Allowed	Allowed	Allowed
PERSISTENT RESERVE OUT	see table 45				
READ ATTRIBUTE	Allowed <sup>b</sup>	Conflict	Allowed	Allowed <sup>b</sup>	Conflict
READ BUFFER	Allowed <sup>b</sup>	Conflict	Allowed	Allowed <sup>b</sup>	Conflict
READ MEDIA SERIAL NUMBER	Allowed	Allowed	Allowed	Allowed	Allowed
RECEIVE CREDENTIAL	Conflict	Conflict	Allowed	Conflict	Conflict
RECEIVE COPY RESULTS	Conflict	Conflict	Allowed	Conflict	Conflict
RECEIVE DIAGNOSTIC RESULTS	Allowed <sup>b</sup>	Conflict	Allowed	Allowed <sup>b</sup>	Conflict
RELEASE(6)/ RELEASE(10)	As defined in SPC-2 <sup>a</sup>				
REPORT ALIASES	Allowed	Allowed	Allowed	Allowed	Allowed
REPORT IDENTIFYING INFORMATION	Allowed	Allowed	Allowed	Allowed	Allowed
REPORT LUNS	Allowed	Allowed	Allowed	Allowed	Allowed
REPORT PRIORITY	Allowed	Allowed	Allowed	Allowed	Allowed
REPORT SUPPORTED OPERATION CODES	Allowed <sup>b</sup>	Conflict	Allowed	Allowed <sup>b</sup>	Conflict
REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS	Allowed <sup>b</sup>	Conflict	Allowed	Allowed <sup>b</sup>	Conflict
Key: <b>Excl</b> =Exclusive, <b>RR</b> =Registrants Only or All Registrants, <> Not Equal					
<sup>a</sup> Exceptions to the behavior of the RESERVE and RELEASE commands described in SPC-2 are defined in 5.7.3. <sup>b</sup> Logical units claiming compliance with previous versions of this standard (e.g., SPC-2, SPC-3) may return RESERVATION CONFLICT in this case. Logical units may report whether certain commands are allowed in ALLOW COMMANDS field of the parameter data returned by the PERSISTENT RESERVE IN command with REPORT CAPABILITIES service action (see 6.13.4).					

Table 44 — SPC commands that are allowed in the presence of various reservations (part 3 of 3)

Command	Addressed logical unit has this type of persistent reservation held by another I_T nexus				
	From any I_T nexus		From registered I_T nexus (RR all types)	From not registered I_T nexus	
	Write Excl	Excl Access		Write Excl RR	Excl Access – RR
REPORT TARGET PORT GROUPS	Allowed	Allowed	Allowed	Allowed	Allowed
REPORT TIMESTAMP	Allowed	Allowed	Allowed	Allowed	Allowed
REQUEST SENSE	Allowed	Allowed	Allowed	Allowed	Allowed
RESERVE(6) / RESERVE(10)	As defined in SPC-2 <sup>a</sup>				
SECURITY PROTOCOL IN	Allowed	Conflict	Allowed	Allowed	Conflict
SECURITY PROTOCOL OUT	Conflict	Conflict	Allowed	Conflict	Conflict
SEND DIAGNOSTIC	Conflict	Conflict	Allowed	Conflict	Conflict
SET IDENTIFYING INFORMATION	Conflict	Conflict	Allowed	Conflict	Conflict
SET PRIORITY	Conflict	Conflict	Allowed	Conflict	Conflict
SET TARGET PORT GROUPS	Conflict	Conflict	Allowed	Conflict	Conflict
SET TIMESTAMP	Conflict	Conflict	Allowed	Conflict	Conflict
TEST UNIT READY	Allowed <sup>b</sup>	Allowed <sup>b</sup>	Allowed	Allowed <sup>b</sup>	Allowed <sup>b</sup>
WRITE ATTRIBUTE	Conflict	Conflict	Allowed	Conflict	Conflict
WRITE BUFFER	Conflict	Conflict	Allowed	Conflict	Conflict
Key: <b>Excl</b> =Exclusive, <b>RR</b> =Registrants Only or All Registrants, <> Not Equal					
<sup>a</sup> Exceptions to the behavior of the RESERVE and RELEASE commands described in SPC-2 are defined in 5.7.3. <sup>b</sup> Logical units claiming compliance with previous versions of this standard (e.g., SPC-2, SPC-3) may return RESERVATION CONFLICT in this case. Logical units may report whether certain commands are allowed in ALLOW COMMANDS field of the parameter data returned by the PERSISTENT RESERVE IN command with REPORT CAPABILITIES service action (see 6.13.4).					

**Table 45 — PERSISTENT RESERVE OUT service actions that are allowed in the presence of various reservations**

Command Service Action	Addressed logical unit has a persistent reservation held by another I_T nexus	
	Command is from a registered I_T nexus	Command is from a not registered I_T nexus
CLEAR	Allowed	Conflict
PREEMPT	Allowed	Conflict
PREEMPT AND ABORT	Allowed	Conflict
REGISTER	Allowed	Allowed
REGISTER AND IGNORE EXISTING KEY	Allowed	Allowed
REGISTER AND MOVE	Conflict	Conflict
RELEASE	Allowed <sup>a</sup>	Conflict
RESERVE	Conflict	Conflict
<sup>a</sup> The reservation is not released (see 5.7.11.2).		

The time at which a reservation is established with respect to other tasks being managed by the device server is vendor specific. Successful completion of a reservation command indicates that the new reservation is established. A reservation may apply to some or all of the tasks in the task set before the completion of the reservation command. The reservation shall apply to all tasks received by the device server after successful completion of the reservation command. Any persistent reserve service action shall be performed as a single indivisible event.

Multiple persistent reserve service actions may be present in the task set at the same time. The order of processing of such service actions is defined by the task set management requirements defined in SAM-4, but each is processed as a single indivisible command without any interleaving of actions that may be required by other reservation commands.

### 5.7.2 Third party persistent reservations

Except for all registrants type reservations, a reservation holder (see 5.7.10) may move the persistent reservation to a third party (e.g., a copy manager supporting the EXTENDED COPY command) using the REGISTER AND MOVE service action (see 5.7.8). A copy manager supporting the EXTENDED COPY command may be instructed to move the persistent reservation to a specified I\_T nexus using the third party persistent reservations source I\_T nexus segment descriptor (see 6.3.7.19).

### 5.7.3 Exceptions to SPC-2 RESERVE and RELEASE behavior

This subclause defines exceptions to the behavior of the RESERVE and RELEASE commands defined in SPC-2. The RESERVE and RELEASE commands are obsolete in this standard, except for the behavior defined in this subclause. Device servers that operate using the exceptions described in this subclause shall set the CRH bit to one in the parameter data returned by the REPORT CAPABILITIES service action of the PERSISTENT RESERVE IN command (see 6.13.4).

A RELEASE(6) or RELEASE(10) command shall complete with GOOD status, but the persistent reservation shall not be released, if the command is received from:

- a) An I\_T nexus that is a persistent reservation holder (see 5.7.10); or
- b) An I\_T nexus that is registered if a registrants only or all registrants type persistent reservation is present.

A RESERVE(6) or RESERVE(10) command shall complete with GOOD status, but no reservation shall be established and the persistent reservation shall not be changed, if the command is received from:

- a) An I\_T nexus that is a persistent reservation holder; or
- b) An I\_T nexus that is registered if a registrants only or all registrants type persistent reservation is present.

In all other cases, a RESERVE(6) command, RESERVE(10) command, RELEASE(6) command, or RELEASE(10) command shall be processed as defined in SPC-2.

#### **5.7.4 Persistent reservations interactions with IKEv2-SCSI SA creation**

If a PERSISTENT RESERVE OUT command is received while an IKEv2-SCSI CCS is in progress (see 5.14.4), the command shall be terminated with CHECK CONDITION status, with the sense key NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS. The sense key specific additional sense data may be set as described in 5.14.5.

#### **5.7.5 Preserving persistent reservations and registrations**

##### **5.7.5.1 Preserving persistent reservations and registrations through power loss**

The application client may request activation of the persist through power loss device server capability to preserve the persistent reservation and registrations across power cycles by setting the APTPL bit to one in the PERSISTENT RESERVE OUT parameter data sent with a REGISTER service action, REGISTER AND IGNORE EXISTING KEY service action, or REGISTER AND MOVE service action.

After the application client enables the persist through power loss capability the device server shall preserve the persistent reservation, if any, and all current and future registrations associated with the logical unit to which the REGISTER service action, the REGISTER AND IGNORE EXISTING KEY service action, or REGISTER AND MOVE service action was addressed until an application client disables the persist through power loss capability. The APTPL value from the most recent successfully completed REGISTER service action, REGISTER AND IGNORE EXISTING KEY service action, or REGISTER AND MOVE service action from any application client shall determine the logical unit's behavior in the event of a power loss.

The device server shall preserve the following information for each existing registration across any hard reset, logical unit reset, or I\_T nexus loss, and if the persist through power loss capability is enabled, across any power cycle:

- a) For SCSI transport protocols where initiator port names (see 3.1.71) are required, the initiator port name; otherwise, the initiator port identifier (see 3.1.70);
- b) Reservation key; and
- c) Indication of the target port to which the registration was applied.

The device server shall preserve the following information about the existing persistent reservation across any hard reset, logical unit reset, or I\_T nexus loss, and if the persist through power loss capability is enabled, across any power cycle:

- a) For SCSI transport protocols where initiator port names are required, the initiator port name; otherwise, the initiator port identifier;
- b) Reservation key;
- c) Scope;
- d) Type; and
- e) Indication of the target port through which the reservation was established.

NOTE 9 - The scope of a persistent reservation is always LU\_SCOPE (see 6.13.3.3). For an all registrants type persistent reservation, only the scope and type need to be preserved.

#### **5.7.5.2 Nonvolatile memory considerations for preserving persistent reservations and registrations**

The capability of preserving persistent reservations and registrations across power cycles requires logical units to use nonvolatile memory within the SCSI device. Any logical unit that supports the persist through power loss capability of persistent reservation and has nonvolatile memory that is not ready shall allow the following commands into the task set:

- a) INQUIRY;
- b) LOG SENSE;
- c) READ BUFFER;
- d) REPORT LUNS;
- e) REQUEST SENSE;
- f) START STOP UNIT (with the start bit set to one and POWER CONDITION field value of 0h); and
- g) WRITE BUFFER.

When nonvolatile memory has not become ready since a power cycle, commands other than those listed in this subclause shall be terminated with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set as described in table 270 (see 6.37).

#### **5.7.6 Finding persistent reservations and reservation keys**

##### **5.7.6.1 Summary of commands for finding persistent reservations and reservation keys**

The application client may obtain information about the persistent reservation and the reservation keys (i.e., registrations) that are present within a device server by issuing a PERSISTENT RESERVE IN command with a READ RESERVATION service action, a READ KEYS service action, or a READ FULL STATUS service action.

##### **5.7.6.2 Reporting reservation keys**

An application client may send a PERSISTENT RESERVE IN command with READ KEYS service action to determine if any I\_T nexuses have been registered with a logical unit through any target port.

In response to a PERSISTENT RESERVE IN with READ KEYS service action the device server shall report the following:

- a) The current PRgeneration value (see 6.13.2); and
- b) The reservation key for every I\_T nexus that is currently registered regardless of the target port through which the registration occurred.



The PRgeneration value allows the application client to verify that the configuration of the I\_T nexuses registered with a logical unit has not been modified.

Duplicate reservation keys shall be reported if multiple I\_T nexuses are registered using the same reservation key.

If an application client uses a different reservation key for each I\_T nexus, the application client may use the reservation key to uniquely identify an I\_T nexus.

#### **5.7.6.3 Reporting the persistent reservation**

An application client may send a PERSISTENT RESERVE IN command with READ RESERVATION service action to receive the persistent reservation information.

In response to a PERSISTENT RESERVE IN command with READ RESERVATION service action the device server shall report the following information for the persistent reservation, if any:

- a) The current PRgeneration value (see 6.13.2);
- b) The registered reservation key, if any, associated with the I\_T nexus that holds the persistent reservation (see 5.7.10). If the persistent reservation is an all registrants type, the registered reservation key reported shall be zero; and
- c) The scope and type of the persistent reservation, if any.

If an application client uses a different reservation key for each I\_T nexus, the application client may use the reservation key to associate the persistent reservation with the I\_T nexus that holds the persistent reservation. This association is done using techniques that are outside the scope of this standard.

#### **5.7.6.4 Reporting full status**

An application client may send a PERSISTENT RESERVE IN command with READ FULL STATUS service action to receive all information about registrations and the persistent reservation, if any.

In response to a PERSISTENT RESERVE IN command with READ FULL STATUS service action the device server shall report the current PRgeneration value (see 6.13.2) and, for every I\_T nexus that is currently registered, the following information:

- a) The registered reservation key;
- b) Whether the I\_T nexus is a persistent reservation holder;
- c) If the I\_T nexus is a persistent reservation holder, the scope and type of the persistent reservation;
- d) The relative target port identifier identifying the target port of the I\_T nexus; and
- e) A TransportID identifying the initiator port of the I\_T nexus.

#### **5.7.7 Registering**

To establish a persistent reservation the application client shall first register an I\_T nexus with the device server. An application client registers with a logical unit by issuing a PERSISTENT RESERVE OUT command with REGISTER service action or REGISTER AND IGNORE EXISTING KEY service action.

If the I\_T nexus has an established registration, an application client may remove the reservation key (see 5.7.11.3). This is accomplished by issuing a PERSISTENT RESERVE OUT command with a REGISTER service action or a REGISTER AND IGNORE EXISTING KEY service action as shown in table 46 and table 47, respectively.

If an I\_T nexus has not yet established a reservation key or the reservation key and registration have been removed, an application client may register that I\_T nexus and zero or more specified unregistered I\_T nexuses by issuing a PERSISTENT RESERVE OUT command with REGISTER service action as defined in table 46.

If the I\_T nexus has an established registration, the application client may change the reservation key by issuing a PERSISTENT RESERVE OUT command with REGISTER service action as defined in table 46.

**Table 46 — Register behaviors for a REGISTER service action**

Command I_T nexus status	Parameter list fields <sup>a</sup>			Results
	RESERVATION KEY	SERVICE ACTION RESERVATION KEY	SPEC_I_PT	
received on an unregistered I_T nexus	zero	zero	ignore	Do nothing except return GOOD status.
		non-zero	zero	Register the I_T nexus on which the command was received with the value specified in the SERVICE ACTION RESERVATION KEY field.
			one	Register the I_T nexus on which the command was received and each unregistered I_T nexus specified in the parameter list with the value specified in the SERVICE ACTION RESERVATION KEY field. <sup>b</sup>
	non-zero	ignore	ignore	Return RESERVATION CONFLICT status.
received on a registered I_T nexus	Not equal to I_T nexus reservation key	ignore	ignore	Return RESERVATION CONFLICT status.
	Equal to I_T nexus reservation key	zero	zero	Unregister the I_T nexus on which the command was received (see 5.7.11.3).
			one	Return CHECK CONDITION status. <sup>c</sup>
		non-zero	zero	Change the reservation key of the I_T nexus on which the command was received to the value specified in the SERVICE ACTION RESERVATION KEY field.
			one	Return CHECK CONDITION status. <sup>c</sup>

<sup>a</sup> For requirements regarding the parameter list fields not shown in this table see 6.14.3.

<sup>b</sup> If any I\_T nexus specified in the parameter list is registered, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST. Devices compliant with previous versions of this standard may return an additional sense code set to INVALID FIELD IN CDB.

<sup>c</sup> The sense key shall be set to ILLEGAL REQUEST, and the additional sense code shall be set to INVALID FIELD IN PARAMETER LIST. Devices compliant with previous versions of this standard may return an additional sense code set to INVALID FIELD IN CDB.

Alternatively, an application client may establish a reservation key for an I\_T nexus without regard for whether one has previously been established by issuing a PERSISTENT RESERVE OUT command with REGISTER AND IGNORE EXISTING KEY service action as defined in table 47.

**Table 47 — Register behaviors for a REGISTER AND IGNORE EXISTING KEY service action**

Command I_T nexus status	Parameter list fields <sup>a</sup>	Results
	SERVICE ACTION RESERVATION KEY	
received on an unregistered I_T nexus	zero	Do nothing except return GOOD status.
	non-zero	Register the I_T nexus on which the command was received with the value specified in the SERVICE ACTION RESERVATION KEY field.
received on a registered I_T nexus	zero	Unregister the I_T nexus on which the command was received (see 5.7.11.3).
	non-zero	Change the reservation key of the I_T nexus on which the command was received to the value specified in the SERVICE ACTION RESERVATION KEY field.
<sup>a</sup> The RESERVATION KEY field is ignored when processing a REGISTER AND IGNORE EXISTING KEY service action. For requirements regarding other parameter list fields not shown in this table see 6.14.3.		

If a PERSISTENT RESERVE OUT command with a REGISTER service action or a REGISTER AND IGNORE EXISTING KEY service action is attempted, but there are insufficient device server resources to complete the operation, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INSUFFICIENT REGISTRATION RESOURCES.

In response to a PERSISTENT RESERVE OUT command with a REGISTER service action or a REGISTER AND IGNORE EXISTING KEY service action the device server shall perform a registration for each specified I\_T nexus by doing the following as an uninterrupted series of actions:

- a) Process the registration request regardless of any persistent reservations;
- b) Process the APTPL bit;
- c) Ignore the contents of the SCOPE and TYPE fields;
- d) Associate the reservation key specified in the SERVICE ACTION RESERVATION KEY field with the I\_T nexus being registered, where:
  - A) The I\_T nexus(es) being registered are shown in table 48; and
  - B) Regardless of how the I\_T nexus initiator port is specified, the association for the initiator port is based on either the initiator port name (see 3.1.71) on SCSI transport protocols where port names are required or the initiator port identifier (see 3.1.70) on SCSI transport protocols where port names are not required;
- e) Register the reservation key specified in the SERVICE ACTION RESERVATION KEY field without changing any persistent reservation that may exist; and

- f) Retain the reservation key specified in the SERVICE ACTION RESERVATION KEY field and associated information.

**Table 48 — I\_T Nexuses being registered**

SPEC_I_PT <sup>a</sup>	ALL_TG_PT	I_T nexus(es) being registered	
		Initiator port	Target port
0	0	The port's names or identifiers to be registered are determined from the I_T nexus on which the PERSISTENT RESERVE OUT command was received	
0	1	The port's name or identifier to be registered is determined from the I_T nexus on which the PERSISTENT RESERVE OUT command was received	Register all of the target ports in the SCSI target device
1	0	a) The port's name or identifier to be registered is determined from the I_T nexus on which the PERSISTENT RESERVE OUT command was received; and b) Specified by each TransportID in the additional parameter data (see 6.14.3)	The port's name or identifier to be registered is determined from the I_T nexus on which the PERSISTENT RESERVE OUT command was received
1	1	a) The port's name or identifier to be registered is determined from the I_T nexus on which the PERSISTENT RESERVE OUT command was received; and b) Specified by each TransportID in the additional parameter data	Register all of the target ports in the SCSI target device
<sup>a</sup> If the SPEC_I_PT bit is set to one and the service action is REGISTER AND IGNORE EXISTING KEY, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.			

After the registration request has been processed, the device server shall then allow other PERSISTENT RESERVE OUT commands from the registered I\_T nexus to be processed. The device server shall retain the reservation key until the key is changed as described in this subclause or removed as described in 5.7.11.

Any PERSISTENT RESERVE OUT command service action received from an unregistered I\_T nexus, other than the REGISTER or the REGISTER AND IGNORE EXISTING KEY service action, shall be completed with RESERVATION CONFLICT status.

It is not an error for an I\_T nexus that is registered to be registered again with the same reservation key or a new reservation key. A registration shall have no effect on any other registrations (e.g., when more than one I\_T nexus is registered with the same reservation key and one of those I\_T nexuses registers again it has no effect on the other I\_T nexus' registrations). A registration that contains a non-zero value in the SERVICE ACTION RESERVATION KEY field shall have no effect on any persistent reservations (i.e., the reservation key for an I\_T nexus may be changed without affecting any previously created persistent reservation).

Multiple I\_T nexuses may be registered with the same reservation key. An application client may use the same reservation key for other I\_T nexuses and logical units.

### 5.7.8 Registering and moving the reservation

The PERSISTENT RESERVE OUT command REGISTER AND MOVE service action is used to register a specified I\_T nexus (see table 49) and move the reservation to establish that I\_T nexus as the reservation holder.

**Table 49 — Register behaviors for a REGISTER AND MOVE service action**

Command I_T nexus status	Parameter list fields <sup>a</sup>			Results
	RESERVATION KEY	SERVICE ACTION RESERVATION KEY	UNREG	
received on an unregistered I_T nexus	ignore	ignore	ignore	Return RESERVATION CONFLICT status. <sup>d</sup>
received on the registered I_T nexus of reservation holder	Not equal to I_T nexus reservation key	ignore	ignore	Return RESERVATION CONFLICT status.
	Equal to I_T nexus reservation key	zero	ignore	Return CHECK CONDITION status. <sup>b</sup>
		non-zero <sup>c</sup>	zero	The I_T nexus on which PERSISTENT RESERVE OUT command was received shall remain registered. See this subclause for the registration and the move specifications.
			one	The I_T nexus on which PERSISTENT RESERVE OUT command was received shall be unregistered (see 5.7.11.3) upon completion of command processing. See this subclause for the registration and the move specifications.
received on a registered I_T nexus that is not the reserva- tion holder	ignore	ignore	ignore	Return RESERVATION CONFLICT status.
<sup>a</sup> For requirements regarding other parameter list fields not shown in this table see 6.14.4. <sup>b</sup> The sense key shall be set to ILLEGAL REQUEST, and the additional sense code shall be set to INVALID FIELD IN PARAMETER LIST. Devices compliant with previous versions of this standard may return an additional sense code set to INVALID FIELD IN CDB. <sup>c</sup> The application client and backup application should use the same reservation key. <sup>d</sup> Devices compliant with previous versions of this standard may return CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.				

If a PERSISTENT RESERVE OUT command with a REGISTER AND MOVE service action is attempted, but there are insufficient device server resources to complete the operation, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INSUFFICIENT REGISTRATION RESOURCES.

If a PERSISTENT RESERVE OUT command with a REGISTER AND MOVE service action is received and the established persistent reservation is a Write Exclusive - All Registrants type or Exclusive Access - All Registrants type reservation, then the command shall be completed with RESERVATION CONFLICT status.

If a PERSISTENT RESERVE OUT command with a REGISTER AND MOVE service action is received and there is no persistent reservation established, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

If a PERSISTENT RESERVE OUT command with a REGISTER AND MOVE service action specifies a TransportID that is the same as the initiator port of the I\_T nexus on which the command received, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

In response to a PERSISTENT RESERVE OUT command with a REGISTER AND MOVE service action the device server shall perform a register and move by doing the following as an uninterrupted series of actions:

- a) Process the APTPL bit;
- b) Ignore the contents of the SCOPE and TYPE fields;
- c) Associate the reservation key specified in the SERVICE ACTION RESERVATION KEY field with the I\_T nexus specified as the destination of the register and move, where:
  - A) The I\_T nexus is specified by the TransportID and the RELATIVE TARGET PORT IDENTIFIER field (see 6.14.4); and
  - B) Regardless of the TransportID format used, the association for the initiator port is based on either the initiator port name (see 3.1.71) on SCSI transport protocols where port names are required or the initiator port identifier (see 3.1.70) on SCSI transport protocols where port names are not required;
- d) Register the reservation key specified in the SERVICE ACTION RESERVATION KEY field;
- e) Retain the reservation key specified in the SERVICE ACTION RESERVATION KEY field and associated information;
- f) Release the persistent reservation for the persistent reservation holder (i.e., the I\_T nexus on which the command was received);
- g) Move the persistent reservation to the specified I\_T nexus using the same scope and type as the persistent reservation released in item f); and
- h) If the UNREG bit is set to one, unregister (see 5.7.11.3) the I\_T nexus on which PERSISTENT RESERVE OUT command was received.

It is not an error for a REGISTER AND MOVE service action to register an I\_T nexus that is already registered with the same reservation key or a different reservation key.

### 5.7.9 Reserving

An application client creates a persistent reservation by issuing a PERSISTENT RESERVE OUT command with RESERVE service action through a registered I\_T nexus with the following parameters:

- a) RESERVATION KEY set to the value of the reservation key that is registered with the logical unit for the I\_T nexus; and
- b) TYPE field and SCOPE field set to the persistent reservation being created.

Only one persistent reservation is allowed at a time per logical unit and that persistent reservation has a scope of LU\_SCOPE.

If the device server receives a PERSISTENT RESERVE OUT command from an I\_T nexus other than a persistent reservation holder (see 5.7.10) that attempts to create a persistent reservation when a persistent reservation already exists for the logical unit, then the command shall be completed with RESERVATION CONFLICT status.

If a persistent reservation holder attempts to modify the type or scope of an existing persistent reservation, the command shall be completed with RESERVATION CONFLICT status.

If the device server receives a PERSISTENT RESERVE OUT command with RESERVE service action where the TYPE field and the SCOPE field contain the same values as the existing type and scope from a persistent reservation holder, it shall not make any change to the existing persistent reservation and shall complete the command with GOOD status.

See 5.7.1 for information on when a persistent reservation takes effect.

#### 5.7.10 Persistent reservation holder

The persistent reservation holder is determined by the type of the persistent reservation as follows:

- a) For a persistent reservation of the type Write Exclusive – All Registrants or Exclusive Access – All Registrants, the persistent reservation holder is any registered I\_T nexus; or
- b) For all other persistent reservation types, the persistent reservation holder is the I\_T nexus:
  - A) For which the reservation was established with a PERSISTENT RESERVE OUT command with RESERVE service action, PREEMPT service action, or PREEMPT AND ABORT service action; or
  - B) To which the reservation was moved by a PERSISTENT RESERVE OUT command with REGISTER AND MOVE service action.

A persistent reservation holder has its reservation key returned in the parameter data from a PERSISTENT RESERVE IN command with READ RESERVATION service action as follows:

- a) For a persistent reservation of the type Write Exclusive – All Registrants or Exclusive Access – All Registrants, the reservation key shall be set to zero; or
- b) For all other persistent reservation types, the reservation key shall be set to the registered reservation key for the I\_T nexus that holds the persistent reservation.

It is not an error for a persistent reservation holder to send a PERSISTENT RESERVE OUT command with RESERVE service action to the reserved logical unit with TYPE and SCOPE fields that match those of the persistent reservation (see 5.7.9).

A persistent reservation holder is allowed to release the persistent reservation using the PERSISTENT RESERVE OUT command with RELEASE service action (see 5.7.11.2).

If the registration of the persistent reservation holder is removed (see 5.7.11.1), the reservation shall be released. If the persistent reservation holder is more than one I\_T nexus, the reservation shall not be released until the registrations for all persistent reservation holder I\_T nexuses are removed.

#### 5.7.11 Releasing persistent reservations and removing registrations

##### 5.7.11.1 Summary of service actions that release persistent reservations and remove registrations

An application client may release or preempt the persistent reservation by issuing one of the following commands through a registered I\_T nexus with the RESERVATION KEY field set to the reservation key value that is registered with the logical unit for that I\_T nexus:

- a) A PERSISTENT RESERVE OUT command with RELEASE service action from a persistent reservation holder (see 5.7.11.2);
- b) A PERSISTENT RESERVE OUT command with PREEMPT service action specifying the reservation key of the persistent reservation holder or holders (see 5.7.11.4);
- c) A PERSISTENT RESERVE OUT command with PREEMPT AND ABORT service action specifying the reservation key of the persistent reservation holder or holders (see 5.7.11.5);
- d) A PERSISTENT RESERVE OUT command with CLEAR service action (see 5.7.11.6); or

- e) If the I\_T nexus is the persistent reservation holder and the persistent reservation is not an all registrants type, then a PERSISTENT RESERVE OUT command with REGISTER service action or REGISTER AND IGNORE EXISTING KEY service action with the SERVICE ACTION RESERVATION KEY field set to zero (see 5.7.11.3).

Table 50 defines processing for a persistent reservation released or preempted by an application client based on the reservation type.

**Table 50 — Processing for a released or preempted persistent reservation**

Reservation Type	Processing
Write Exclusive – Registrants Only or Exclusive Access – Registrants Only	When the persistent reservation holder (see 5.7.10) of this reservation type becomes unregistered the persistent reservation shall be released.
Write Exclusive – All Registrants or Exclusive Access – All Registrants	This persistent reservation shall be released when the registration for the last registered I_T nexus is removed or when the type or scope is changed.
Write Exclusive or Exclusive Access	When the persistent reservation holder of this reservation type becomes unregistered the persistent reservation shall be released.

An application client may remove registrations by issuing one of the following commands through a registered I\_T nexus with the RESERVATION KEY field set to the reservation key value that is registered with the logical unit for that I\_T nexus:

- a) A PERSISTENT RESERVE OUT command with PREEMPT service action with the SERVICE ACTION RESERVATION KEY field set to the reservation key (see 5.7.11.4) to be removed;
- b) A PERSISTENT RESERVE OUT command with PREEMPT AND ABORT service action with the SERVICE ACTION RESERVATION KEY field set to the reservation key (see 5.7.11.5) to be removed;
- c) A PERSISTENT RESERVE OUT command with CLEAR service action (see 5.7.11.6); or
- d) A PERSISTENT RESERVE OUT command with REGISTER service action or REGISTER AND IGNORE EXISTING KEY service action with the SERVICE ACTION RESERVATION KEY field set to zero (see 5.7.11.3).

When a reservation key (i.e., registration) has been removed, no information shall be reported for that unregistered I\_T nexus in subsequent READ KEYS service actions until the I\_T nexus is registered again (see 5.7.7).

If the persist through power loss capability is not enabled, loss of power also causes persistent reservations to be released and registrations to be removed. When the most recent APTPL value received by the device server is zero (see 6.14.3), a power cycle:

- a) Releases all persistent reservations; and
- b) Removes all registered reservation keys (see 5.7.7).



### 5.7.11.2 Releasing

Only the persistent reservation holder (see 5.7.10) is allowed to release a persistent reservation.

An application client releases the persistent reservation by issuing a PERSISTENT RESERVE OUT command with RELEASE service action through an I\_T nexus that is a persistent reservation holder with the following parameters:

- a) RESERVATION KEY field set to the value of the reservation key that is registered with the logical unit for the I\_T nexus; and
- b) TYPE field and SCOPE field set to match the persistent reservation being released.

In response to a persistent reservation release request from the persistent reservation holder the device server shall perform a release by doing the following as an uninterrupted series of actions:

- a) Release the persistent reservation;
- b) Not remove any registration(s);
- c) If the released persistent reservation is a registrants only type or all registrants type persistent reservation, the device server shall establish a unit attention condition for the initiator port associated with every registered I\_T nexus other than I\_T nexus on which the PERSISTENT RESERVE OUT command with RELEASE service action was received, with the additional sense code set to RESERVATIONS RELEASED; and
- d) If the persistent reservation is of any other type, the device server shall not establish a unit attention condition.

The established persistent reservation shall not be altered and the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID RELEASE OF PERSISTENT RESERVATION, for a PERSISTENT RESERVE OUT command that specifies the release of a persistent reservation if:

- a) The requesting I\_T nexus is a persistent reservation holder (see 5.7.10); and
- b) The SCOPE and TYPE fields do not match the scope and type of the established persistent reservation.

If there is no persistent reservation or in response to a persistent reservation release request from a registered I\_T nexus that is not a persistent reservation holder (see 5.7.10), the device server shall do the following:

- a) Not release the persistent reservation, if any;
- b) Not remove any registrations; and
- c) Complete the command with GOOD status.

### 5.7.11.3 Unregistering

An application client may remove a registration for an I\_T nexus by issuing a PERSISTENT RESERVE OUT command with REGISTER service action or a REGISTER AND IGNORE EXISTING KEY service action with the SERVICE ACTION RESERVATION KEY field set to zero through that I\_T nexus.

If the I\_T nexus is a reservation holder, the persistent reservation is of an all registrants type, and the I\_T nexus is the last remaining registered I\_T nexus, then the device server shall also release the persistent reservation.

If the I\_T nexus is the reservation holder and the persistent reservation is of a type other than all registrants, the device server shall also release the persistent reservation. If the persistent reservation is a registrants only type, the device server shall establish a unit attention condition for the initiator port associated with every registered I\_T nexus except for the I\_T nexus on which the PERSISTENT RESERVE OUT command was received, with the additional sense code set to RESERVATIONS RELEASED.

#### 5.7.11.4 Preempting

##### 5.7.11.4.1 Overview

A PERSISTENT RESERVE OUT command with PREEMPT service action or PREEMPT AND ABORT service action is used to:

- a) Preempt (i.e., replace) the persistent reservation and remove registrations; or
- b) Remove registrations.

Table 51 lists the actions taken based on the current persistent reservation type and the SERVICE ACTION RESERVATION KEY field in the PERSISTENT RESERVE OUT command.

**Table 51 — Preempting actions**

Reservation Type	Service Action Reservation Key	Action	Reference
All Registrants	Zero	Preempt the persistent reservation and remove registrations.	5.7.11.4.3
	Not Zero	Remove registrations.	5.7.11.4.4
All other types	Zero	Terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.	
	Reservation holder's reservation key	Preempt the persistent reservation and remove registrations.	5.7.11.4.3
	Any other, non-zero reservation key	Remove registrations.	5.7.11.4.4

See figure 7 for a description of how a device server interprets a PREEMPT service action to determine its actions (e.g., preempt the persistent reservation, remove registration, or both preempt the persistent reservation and remove registration).

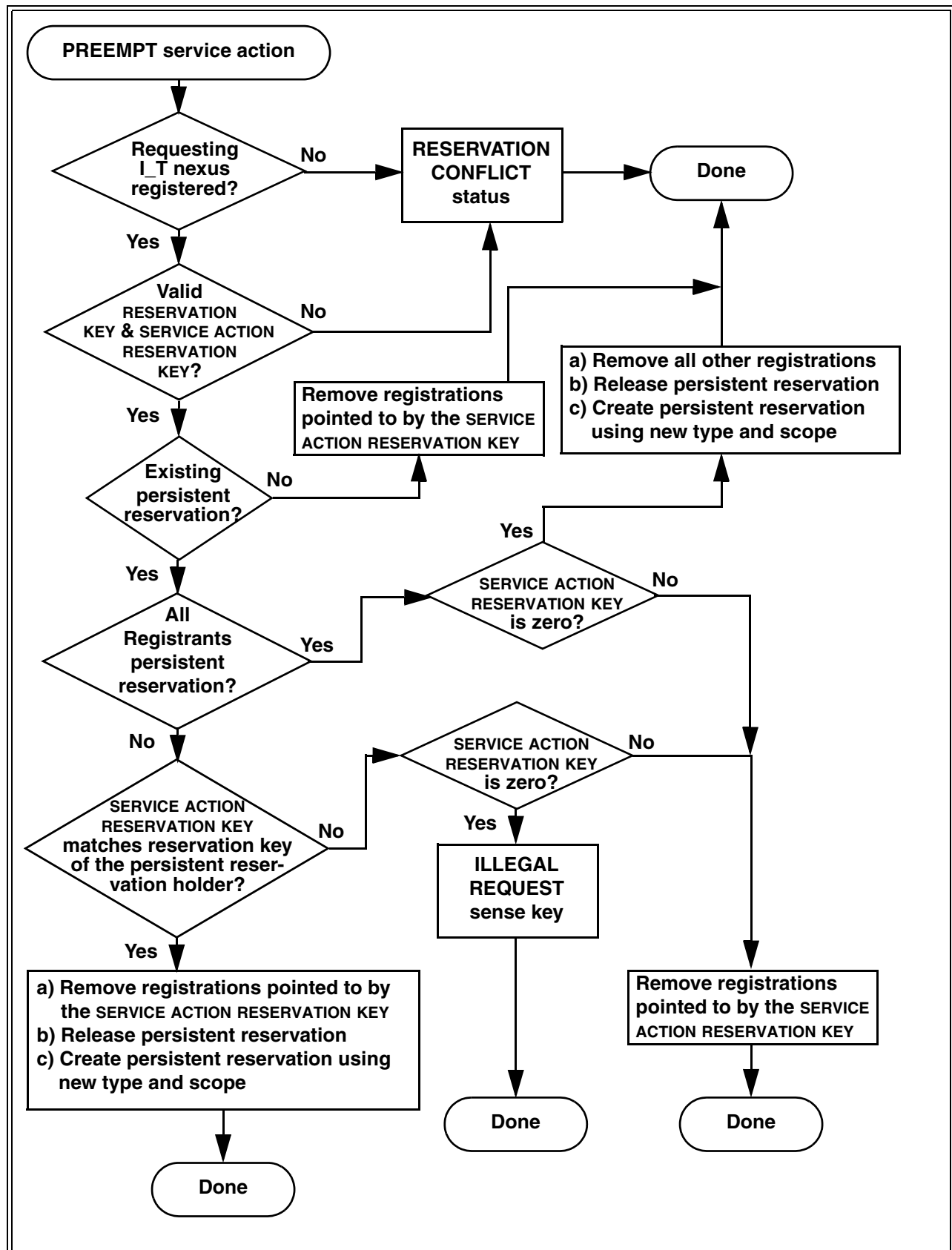


Figure 7 — Device server interpretation of PREEMPT service action

#### 5.7.11.4.2 Failed persistent reservation preempt

If the preempting I\_T nexus' PREEMPT service action or PREEMPT AND ABORT service action fails (e.g., repeated TASK SET FULL status, repeated BUSY status, SCSI transport protocol time-out, or time-out due to the task set being blocked due to failed initiator port or failed SCSI initiator device), the application client may send a LOGICAL UNIT RESET task management function to the failing logical unit to remove blocking tasks and then resend the preempting service action.

#### 5.7.11.4.3 Preempting persistent reservations and registration handling

An application client may preempt the persistent reservation with another persistent reservation by issuing a PERSISTENT RESERVE OUT command with PREEMPT service action or PREEMPT AND ABORT service action through a registered I\_T nexus with the following parameters:

- a) RESERVATION KEY field set to the value of the reservation key that is registered with the logical unit for the I\_T nexus;
- b) SERVICE ACTION RESERVATION KEY field set to the value of the reservation key of the persistent reservation to be preempted; and
- c) TYPE field and SCOPE field set to define a new persistent reservation. The SCOPE and TYPE of the persistent reservation created by the preempting I\_T nexus may be different than those of the persistent reservation being preempted.

If the SERVICE ACTION RESERVATION KEY field identifies a persistent reservation holder (see 5.7.10), the device server shall perform a preempt by doing the following as an uninterrupted series of actions:

- a) Release the persistent reservation for the holder identified by the SERVICE ACTION RESERVATION KEY field;
- b) Remove the registrations for all I\_T nexuses identified by the SERVICE ACTION RESERVATION KEY field, except the I\_T nexus that is being used for the PERSISTENT RESERVE OUT command. If an all registrants persistent reservation is present and the SERVICE ACTION RESERVATION KEY field is set to zero, then all registrations shall be removed except for that of the I\_T nexus that is being used for the PERSISTENT RESERVE OUT command;
- c) Establish a persistent reservation for the preempting I\_T nexus using the contents of the SCOPE and TYPE fields;
- d) Process tasks as defined in 5.7.1;
- e) Establish a unit attention condition for the initiator port associated with every I\_T nexus that lost its persistent reservation and/or registration, with the additional sense code set to REGISTRATIONS PREEMPTED; and
- f) If the type or scope has changed, then for every I\_T nexus whose reservation key was not removed, except for the I\_T nexus on which the PERSISTENT RESERVE OUT command was received, the device server shall establish a unit attention condition for the initiator port associated with that I\_T nexus, with the additional sense code set to RESERVATIONS RELEASED. If the type or scope have not changed, then no unit attention condition(s) shall be established for this reason.

After the PERSISTENT RESERVE OUT command has been completed with GOOD status, new tasks are subject to the persistent reservation restrictions established by the preempting I\_T nexus.

The following tasks shall be subjected in a vendor specific manner either to the restrictions established by the persistent reservation being preempted or to the restrictions established by the preempting I\_T nexus:

- a) A task received after the arrival, but before the completion of the PERSISTENT RESERVE OUT command with the PREEMPT service action or the PREEMPT AND ABORT service action; or
- b) A task in the dormant, blocked, or enabled state (see SAM-4) at the time the PERSISTENT RESERVE OUT command with the PREEMPT service action or the PREEMPT AND ABORT service action is received.

Completion status shall be returned for each task unless it was aborted by a PERSISTENT RESERVE OUT command with the PREEMPT AND ABORT service action and TAS bit set to zero in the Control mode page (see 7.4.6).

If an all registrants persistent reservation is not present, it is not an error for the persistent reservation holder to preempt itself (i.e., a PERSISTENT RESERVE OUT with a PREEMPT service action or a PREEMPT AND ABORT service action with the SERVICE ACTION RESERVATION KEY value equal to the persistent reservation holder's reservation key that is received from the persistent reservation holder). In that case, the device server shall establish the new persistent reservation and maintain the registration.

#### 5.7.11.4.4 Removing registrations

When a registered reservation key does not identify a persistent reservation holder (see 5.7.10), an application client may remove the registration(s) without affecting any persistent reservations by issuing a PERSISTENT RESERVE OUT command with PREEMPT service action through a registered I\_T nexus with the following parameters:

- a) RESERVATION KEY field set to the value of the reservation key that is registered for the I\_T nexus; and
- b) SERVICE ACTION RESERVATION KEY field set to match the reservation key of the registration or registrations being removed.

If the SERVICE ACTION RESERVATION KEY field does not identify a persistent reservation holder or there is no persistent reservation holder (i.e., there is no persistent reservation), then the device server shall perform a preempt by doing the following in an uninterrupted series of actions:

- a) Remove the registrations for all I\_T nexuses specified by the SERVICE ACTION RESERVATION KEY field;
- b) Ignore the contents of the SCOPE and TYPE fields;
- c) Process tasks as defined in 5.7.1; and
- d) Establish a unit attention condition for the initiator port associated with every I\_T nexus that lost its registration other than the I\_T nexus on which the PERSISTENT RESERVE OUT command was received, with the additional sense code set to REGISTRATIONS PREEMPTED.

If a PERSISTENT RESERVE OUT with a PREEMPT service action or a PREEMPT AND ABORT service action sets the SERVICE ACTION RESERVATION KEY field to a value that does not match any registered reservation key, then the device server shall complete the command with RESERVATION CONFLICT status.

It is not an error for a PERSISTENT RESERVE OUT with a PREEMPT service action or a PREEMPT AND ABORT service action to set the RESERVATION KEY and the SERVICE ACTION RESERVATION KEY to the same value, however, no unit attention condition is established for the I\_T nexus on which the PERSISTENT RESERVE OUT command was received. The registration is removed.

#### 5.7.11.5 Preempting and aborting

The application client's request for and the device server's responses to a PERSISTENT RESERVE OUT command PREEMPT AND ABORT service action are identical to the responses to a PREEMPT service action (see 5.7.11.4) except for the additions described in this subclause. If no reservation conflict occurred, the device server shall perform the following uninterrupted series of actions:

- a) If the persistent reservation is not an all registrants type then:
  - A) If the TST field is 000b (see 7.4.6) and the faulted I\_T nexus (see 3.1.53), if any, is not the I\_T nexus associated with the persistent reservation or registration being preempted, then the task set ACA condition shall be processed as defined in SAM-4;

- B) If the TST field contains 000b and the faulted I\_T nexus, if any, is the I\_T nexus associated with the persistent reservation or registration being preempted, then the PERSISTENT RESERVE OUT command shall be processed without regard for the task set ACA condition; or
- C) If the TST field contains 001b, then the ACA condition shall be processed as defined in SAM-4;
- b) Perform the uninterrupted series of actions described for the PREEMPT service action (see 5.7.11.4);
- c) All tasks from the I\_T nexus(es) associated with the persistent reservations or registrations being preempted (i.e., preempted tasks) except the task containing the PERSISTENT RESERVE OUT command itself shall be aborted as defined in SAM-4. If an aborted task is a command that causes the device server to generate additional commands and data transfers (e.g., EXTENDED COPY), then all commands and data transfers generated by the command shall be aborted before the ABORT TASK SET task management function is considered completed. After the ABORT TASK SET function has completed, all new tasks are subject to the persistent reservation restrictions established by the preempting I\_T nexus;
- d) If the persistent reservation is not an all registrants type, then the device server shall clear any ACA condition associated with an I\_T nexus being preempted and shall abort any tasks with an ACA attribute received on that I\_T nexus;
- e) If the persistent reservation is an all registrants type, then:
  - A) If the service action reservation key is set to zero, the device server shall clear any ACA condition and shall abort any tasks with an ACA attribute; or
  - B) If the service action reservation key is not set to zero, the device server shall do the following for any I\_T nexus registered using the specified reservation key:
    - a) Clear any ACA condition; and
    - b) Abort any tasks with an ACA attribute;
- and
- f) For logical units that implement the PREVENT ALLOW MEDIUM REMOVAL command (see SBC-3, SSC-3, and SMC-3), the device server shall perform an action equivalent to the processing of a PREVENT ALLOW MEDIUM REMOVAL command with the PREVENT field equal to zero received on the I\_T nexuses associated with the persistent reservation being preempted.

The actions described in this subclause shall be performed for all I\_T nexuses that are registered with the non-zero SERVICE ACTION RESERVATION KEY value, without regard for whether the preempted I\_T nexuses hold the persistent reservation. If the SERVICE ACTION RESERVATION KEY value is zero and an all registrants persistent reservation is present, the device server shall abort all tasks for all registered I\_T nexuses.

#### 5.7.11.6 Clearing

Any application client may release the persistent reservation and remove all registrations from a device server by issuing a PERSISTENT RESERVE OUT command with CLEAR service action through a registered I\_T nexus with the following parameter:

- a) RESERVATION KEY field set to the value of the reservation key that is registered with the logical unit for the I\_T nexus.

In response to this request the device server shall perform a clear by doing the following as part of an uninterrupted series of actions:

- a) Release the persistent reservation, if any;
- b) Remove all registration(s) (see 5.7.7);
- c) Ignore the contents of the SCOPE and TYPE fields;
- d) Continue normal processing of any tasks from any I\_T nexus that have been accepted by the device server as allowed (i.e., nonconflicting); and
- e) Establish a unit attention condition for the initiator port associated with every registered I\_T nexus other than the I\_T nexus on which the PERSISTENT RESERVE OUT command with CLEAR service action was received, with the additional sense code set to RESERVATIONS PREEMPTED.

NOTE 10 - Application clients should not use the CLEAR service action except during recovery operations that are associated with a specific initiator port, since the effect of the CLEAR service action defeats the persistent reservations features that protect data integrity.

## 5.8 Multiple target port and initiator port behavior

SAM-4 specifies the behavior of logical units being accessed by application clients through more than one initiator port and/or through more than one target port. Additional initiator ports and target ports allow the definition of multiple I\_T nexuses through which the device server may be reached. Multiple I\_T nexuses may be used to improve the availability of logical units in the presence of certain types of failures and to improve the performance between an application client and logical unit when some I\_T nexuses may be busy.

If one target port is being used by an initiator port, accesses attempted through other target port(s) may:

- a) Receive a status of BUSY; or
- b) Be accepted as if the other target port(s) were not in use.

The device server shall indicate the presence of multiple target ports by setting the MULTIP bit to one in its standard INQUIRY data.

Only the following operations allow one I\_T nexus to interact with the tasks of other I\_T nexuses:

- a) The PERSISTENT RESERVE OUT with PREEMPT service action preempts persistent reservations (see 5.7.11.4);
- b) The PERSISTENT RESERVE OUT with CLEAR service action releases persistent reservations for all I\_T nexuses (see 5.7.11.6); and
- c) Commands and task management functions that allow one I\_T nexus to abort tasks received on a different I\_T nexus (see SAM-4).

## 5.9 Target port group access states

### 5.9.1 Target port group access overview

Logical units may be connected to one or more service delivery subsystems via multiple target ports (see SAM-4). The access to logical units through the multiple target ports may be symmetrical (see 5.9.3) or asymmetrical (see 5.9.2).

### 5.9.2 Asymmetric logical unit access

#### 5.9.2.1 Introduction to asymmetric logical unit access

Asymmetric logical unit access occurs when the access characteristics of one port may differ from those of another port. SCSI target devices with target ports implemented in separate physical units may need to designate differing levels of access for the target ports associated with each logical unit. While commands and task management functions (see SAM-4) may be routed to a logical unit through any target port, the performance may not be optimal, and the allowable command set may be less complete than when the same commands and task management functions are routed through a different target port. In addition, some target ports may be in a state (e.g., offline) that is unique to that target port. When a failure on the path to one target port is detected, the SCSI target device may perform automatic internal reconfiguration to make a logical unit accessible from a different set of target ports or may be instructed by the application client to make a logical unit accessible from a different set of target ports.

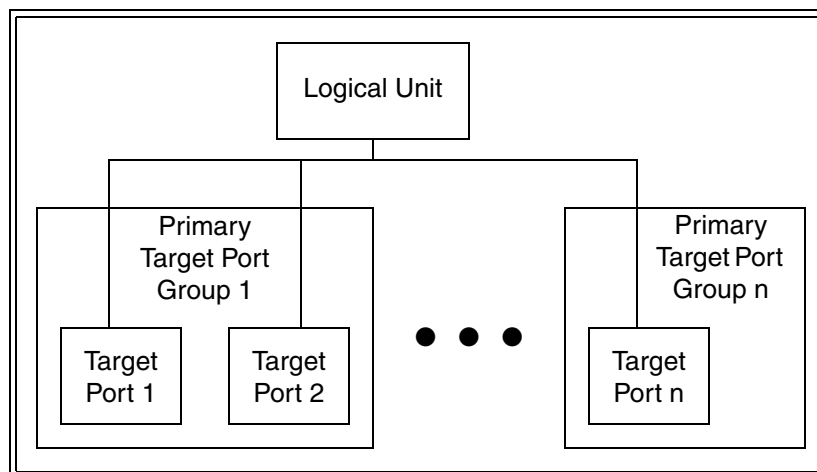
A target port characteristic called primary target port asymmetric access state (see 5.9.2.4) defines properties of a target port and the allowable command set for a logical unit when commands and task management functions are routed through the target port maintaining that state.

A primary target port group is defined as a set of target ports that are in the same primary target port asymmetric access state at all times (i.e., a change in one target port's primary target port asymmetric access state implies an equivalent change in the primary target port asymmetric access state of all target ports in the same primary target port group). A primary target port group asymmetric access state is defined as the primary target port asymmetric access state common to the set of target ports in a primary target port group. One target port is a member of at most one primary target port group for a logical unit group (see 7.7.3.9). The grouping of target ports in a primary target port group is vendor specific.

A logical unit may have commands and task management functions routed through multiple primary target port groups. Logical units support asymmetric logical unit access if different primary target port groups may be in different primary target port group asymmetric access states.

An example of asymmetric logical unit access is a SCSI controller device with two separated controllers where all target ports on one controller are in the same primary target port asymmetric access state with respect to a logical unit and are members of the same primary target port group. Target ports on the other controller are members of another primary target port group. The behavior of each primary target port group may be different with respect to a logical unit, but all members of a single primary target port group are always in the same primary target port group asymmetric access state with respect to a logical unit.

An example of primary target port groups is shown in figure 8.



**Figure 8 — Primary target port group example**

Another target port characteristic called secondary target port asymmetric access state (see 5.9.2.4) indicates a condition that affects the way in which an individual target port participates in its assigned primary target port group. All target ports, if any, in one secondary target port asymmetric access state are grouped into a secondary target port group. Secondary target port groups have the following properties:

- a) A target port in any secondary target port group also shall be in one primary target port group;
- b) A change of secondary target port asymmetric access state for one target port shall not cause changes in the secondary target port asymmetric access state of other target ports, if any, in the same secondary target port group; and
- c) A target port may be a member of zero or more secondary target port groups.



The term target port asymmetric access state represents both primary target port asymmetric access states and secondary target port asymmetric access states. The term target port group represents both primary target port groups and secondary target port groups.

#### **5.9.2.2 Explicit and implicit asymmetric logical unit access**

Asymmetric logical unit access may be managed explicitly by an application client using the REPORT TARGET PORT GROUPS (see 6.27) and SET TARGET PORT GROUPS (see 6.35) commands.

Alternatively, asymmetric logical unit access may be managed implicitly by the SCSI target device based on the type of transactions being routed through each target port and the internal configuration capabilities of the primary target port group(s) through which the logical unit may be accessed. The logical units may attempt to maintain full performance across the primary target port groups that are busiest and that show the most reliable performance, allowing other primary target port groups to select a lower performance primary target port asymmetric access state.

Implicit management of secondary target port asymmetric access states is based on the condition of an individual target port and how such conditions affect that target port's ability to participate in its assigned primary target port group.

If both explicit and implicit asymmetric logical unit access management methods are implemented, the precedence of one over the other is vendor specific.

#### **5.9.2.3 Discovery of asymmetric logical unit access behavior**

SCSI logical units with asymmetric logical unit access may be identified using the INQUIRY command. The value in the target port group support (TPGS) field (see 6.4.2) indicates whether or not the logical unit supports asymmetric logical unit access and if so whether implicit or explicit management is supported. The target port asymmetric access states supported by a logical unit may be determined by the REPORT TARGET PORT GROUPS command parameter data (see 6.27).

#### **5.9.2.4 Target port asymmetric access states**

##### **5.9.2.4.1 Target port asymmetric access states overview**

For all SCSI target devices that report in the INQUIRY data that they support asymmetric logical unit access, all of the target ports in a primary target port group (see 5.9.2.1) shall be in the same primary target port asymmetric access state with respect to the ability to route information to a logical unit. The primary target port asymmetric access states are:

- a) Active/optimized;
- b) Active/non-optimized;
- c) Standby; and
- d) Unavailable.

Individual target ports may be in secondary target port groups (see 5.9.2.1) that have the following secondary target port asymmetric access states:

- a) Offline.

##### **5.9.2.4.2 Active/optimized state**

The active/optimized state is a primary target port asymmetric access state (see 3.1.108). When commands and task management functions are being routed through a target port in the active/optimized primary target port

asymmetric access state, the device server shall function (e.g., respond to commands) as specified in the appropriate command standards (see 3.1.27). All target ports within a primary target port group should be capable of immediately accessing the logical unit.

The SCSI target device shall participate in all task management functions as defined in SAM-4 and modified by the applicable SCSI transport protocol standards (see 3.1.140).

#### **5.9.2.4.3 Active/non-optimized state**

The active/non-optimized state is a primary target port asymmetric access state (see 3.1.108). When commands and task management functions are being routed through a target port in the active/non-optimized primary target port asymmetric access state, the device server shall function as specified in the appropriate command standards.

The processing of some task management functions and commands, especially those involving data transfer or caching, may operate with lower performance than they would if the target port were in the active/optimized primary target port asymmetric access state.

The SCSI target device shall participate in all task management functions as defined in SAM-4 and modified by the applicable SCSI transport protocol standards (see 3.1.140).

#### **5.9.2.4.4 Standby state**

The standby state is a primary target port asymmetric access state (see 3.1.108). When being accessed through a target port in the standby primary target port asymmetric access state, the device server shall support those of the following commands that it supports while in the active/optimized primary target port asymmetric access state:

- a) INQUIRY;
- b) LOG SELECT;
- c) LOG SENSE;
- d) MODE SELECT;
- e) MODE SENSE;
- f) REPORT LUNS;
- g) RECEIVE DIAGNOSTIC RESULTS;
- h) SEND DIAGNOSTIC;
- i) REPORT TARGET PORT GROUPS;
- j) SET TARGET PORT GROUPS;
- k) REQUEST SENSE;
- l) PERSISTENT RESERVE IN;
- m) PERSISTENT RESERVE OUT;
- n) Echo buffer modes of READ BUFFER; and
- o) Echo buffer modes of WRITE BUFFER.

The device server may support other commands.

For those commands that are not supported, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT ACCESSIBLE, TARGET PORT IN STANDBY STATE.

The SCSI target device shall participate in all task management functions as defined in SAM-4 and modified by the applicable SCSI transport protocol standards (see 3.1.140).

#### 5.9.2.4.5 Unavailable state

The unavailable state is a primary target port asymmetric access state (see 3.1.108). When being accessed through a target port in the unavailable primary target port asymmetric access state, the device server shall accept only a limited set of commands. The unavailable primary target port asymmetric access state is intended for situations when the target port accessibility to a logical unit may be severely restricted due to SCSI target device limitations (e.g., hardware errors). Therefore it may not be possible to transition from this state to either the active/optimized, active/non-optimized or standby states. The unavailable primary target port asymmetric access state is also intended for minimizing any disruption when using the downloading microcode mode of the WRITE BUFFER command.

While in the unavailable primary target port asymmetric access state, the device server shall support those of the following commands that it supports while in the active/optimized state:

- a) INQUIRY (the peripheral qualifier (see 6.4.2) shall be set to 001b);
- b) REPORT LUNS;
- c) REPORT TARGET PORT GROUPS;
- d) SET TARGET PORT GROUPS;
- e) REQUEST SENSE;
- f) Echo buffer modes of READ BUFFER;
- g) Echo buffer modes of WRITE BUFFER; and
- h) Download microcode mode of WRITE BUFFER.

The device server may support other commands.

For those commands that are not supported, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT ACCESSIBLE, TARGET PORT IN UNAVAILABLE STATE.

The SCSI target device is not required to participate in all task management functions (see SAM-4 and the applicable SCSI transport protocol standards).

#### 5.9.2.4.6 Offline state

The offline state is a secondary target port asymmetric access state (see 3.1.141). Target ports in the offline secondary target port asymmetric access state are not accessible via the service delivery subsystem (e.g. during maintenance, port replacement, port disabled, hot swap, or power failures impacting only some target port). While in the offline secondary target port asymmetric state, the target port is not capable of receiving or responding to any commands or task management functions.

The offline secondary target port asymmetric access state allows a device server to report that some target ports are not capable of being accessed.

After access to the service delivery subsystem is enabled, the target port shall transition out of the offline secondary target port asymmetric access state.

#### 5.9.2.5 Transitions between target port asymmetric access states

The movement from one target port asymmetric access state to another is called a transition.

During a transition between target port asymmetric access states the device server shall respond to a command in one of the following ways:

- a) If during the transition the logical unit is inaccessible, then the transition is performed as a single indivisible event and the device server shall respond by either returning BUSY status, or returning CHECK CONDITION status, with the sense key set to NOT READY, and an the sense code set to LOGICAL UNIT NOT ACCESSIBLE, ASYMMETRIC ACCESS STATE TRANSITION; or
- b) If during the transition the target ports in a primary target port group are able to access the requested logical unit, then the device server shall support those of the following commands that it supports while in the active/optimized primary target port asymmetric access state:
  - A) INQUIRY;
  - B) REPORT LUNS;
  - C) REPORT TARGET PORT GROUPS;
  - D) REQUEST SENSE;
  - E) Echo Buffer modes of READ BUFFER; and
  - F) Echo Buffer modes of WRITE BUFFER.

The device server may support other commands when those commands are routed through a target port that is transitioning between primary target port asymmetric access states.

For those commands that are not supported during a transition, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT ACCESSIBLE, ASYMMETRIC ACCESS STATE TRANSITION.

The SCSI target device is not required to participate in all task management functions.

If the transition was explicit to a supported target port asymmetric access state and it failed, then the command shall be terminated with CHECK CONDITION status, with the sense key set to HARDWARE ERROR, and the additional sense code set to SET TARGET PORT GROUPS COMMAND FAILED. If the transition was to a primary target port asymmetric access state, the primary target port group that encountered the error should complete a transition to the unavailable primary target port asymmetric access state.

If a target port asymmetric access state change occurred as a result of the failed transition, then the device server shall establish a unit attention condition for the initiator port associated with every I\_T nexus other than the I\_T nexus on which the SET TARGET PORT GROUPS command was received with the additional sense code set to ASYMMETRIC ACCESS STATE CHANGED.

If the transition was implicit and it failed, then the device server shall establish a unit attention condition for the initiator port associated with every I\_T nexus with the additional sense code set to IMPLICIT ASYMMETRIC ACCESS STATE TRANSITION FAILED.

An implicit CLEAR TASK SET task management function may be performed following a transition failure.

Once a transition is completed, the new target port asymmetric access state may apply to some or all tasks entered into the task set before the completion of the transition. The new target port asymmetric access state shall apply to all tasks received by the device server after completion of a transition.

If a transition is to the offline secondary target port asymmetric access state, communication with the service delivery subsystem shall be terminated. This may result in commands being terminated and may cause command timeouts to occur on the initiator.

After an implicit target port asymmetric access state change, a device server shall establish a unit attention condition for the initiator port associated with every I\_T nexus with the additional sense code set to ASYMMETRIC ACCESS STATE CHANGED.

After an explicit target port asymmetric access state change, a device server shall establish a unit attention condition with the additional sense code set to ASYMMETRIC ACCESS STATE CHANGED for the initiator port associated with every I\_T nexus other than the I\_T nexus on which the SET TARGET GROUPS command was received.

#### 5.9.2.6 Preference Indicator

A device server may indicate one or more primary target port groups is a preferred primary target port group for accessing a logical unit by setting the PREF bit to one in the target port group descriptor (see 6.27). The preference indication is independent of the primary target port asymmetric access state.

An application client may use the PREF bit value in the target port group descriptor to influence the path selected to a logical unit (e.g., a primary target port group in the standby primary target port asymmetric access state with the PREF bit set to one may be chosen over a primary target port group in the active/optimized primary target port asymmetric access state with the PREF bit set to zero).

The value of the PREF bit for a primary target port group may change whenever an primary target port asymmetric access state changes.

#### 5.9.2.7 Implicit asymmetric logical units access management

SCSI target devices with implicit asymmetric logical units access management are capable using mechanisms other than the SET TARGET PORT GROUPS command to set the:

- a) Primary target port asymmetric access state of a primary target port group; or
- b) Secondary target port asymmetric access state of a target port that is a member of a primary target port group.

All logical units that report in the standard INQUIRY data (see 6.4.2) that they support asymmetric logical units access and support implicit asymmetric logical unit access (i.e., the TPGS field contains 01b or 11b):

- a) Shall implement the INQUIRY command Device Identification VPD page designator types 4h (see 7.7.3.7) and 5h (see 7.7.3.8);
- b) Shall support the REPORT TARGET PORT GROUPS command as described in 6.27; and
- c) May implement the INQUIRY command Device Identification VPD page designator type 6h (see 7.7.3.9).

Implicit logical unit access state changes between primary target port asymmetric access states (see 3.1.108) may be disabled with the IALUAE bit in the Control Extension mode page (see 7.4.7).

#### 5.9.2.8 Explicit asymmetric logical units access management

All logical units that report in the standard INQUIRY data (see 6.4.2) that they support asymmetric logical units access and support explicit asymmetric logical unit access (i.e., the TPGS field contains 10b or 11b):

- a) Shall implement the INQUIRY command Device Identification VPD page (see 7.7.3) designator types 4h and 5h;
- b) Shall support the REPORT TARGET PORT GROUPS command as described in 6.27;
- c) Shall support the SET TARGET PORT GROUPS command as described in 6.35; and
- d) May implement the INQUIRY command Device Identification VPD page designator type 6h (see 7.7.3.9).

### 5.9.2.9 Behavior after power on, hard reset, logical unit reset, and I\_T nexus loss

For all SCSI target devices that report in the standard INQUIRY data (see 6.4.2) that they support only explicit asymmetric logical unit access (i.e., the TPGS field contains 10b), the target port shall preserve the primary target port asymmetric access state during any power on, hard reset, logical unit reset, and I\_T nexus loss.

### 5.9.2.10 Behavior of target ports that are not accessible from the service delivery subsystem

If the offline secondary target port asymmetric access state is supported and a subset of the target ports in a primary target port group are not accessible via the service delivery subsystem (e.g. power failure), then those ports may be reported in a primary target port group consistent with their primary target port asymmetric access state and in the secondary target port group with the offline secondary target port asymmetric access state.

## 5.9.3 Symmetric logical unit access

A device server that provides symmetrical access to a logical unit may use a subset of the asymmetrical logical access features (see 5.9.2) to indicate this ability to an application client, providing an application client a common set of commands to determine how to manage target port access to a logical unit.

Symmetrical logical unit access should be represented as follows:

- a) The TPGS field in the standard INQUIRY data (see 6.4.2) indicates that implicit asymmetric access is supported;
- b) The REPORT TARGET PORT GROUPS command is supported; and
- c) The REPORT TARGET PORT GROUPS parameter data indicates that the same state (e.g., active/optimized state) is in effect for all primary target port groups.

## 5.10 Power conditions

### 5.10.1 Power conditions overview

The optional Power Condition mode page (see 7.4.12) allows an application client to control the power condition of a logical unit in a manner that may reduce power consumption of the SCSI target device. This control is invoked by enabling and setting the idle condition timer and/or the standby condition timer using the mode page. A change in the power condition of any logical unit in a SCSI target device may result in a change in the SCSI target device's power consumption.

In addition to the Power Condition mode page, the power condition of a logical unit may be controlled by the START STOP UNIT command (see SBC-3 or RBC). If both the Power Condition mode page and the START STOP UNIT command methods are being used to control the power condition of the same logical unit, then any START STOP UNIT command's power condition specification shall override the Power Condition mode page's power control and may disable the idle condition and standby condition timers.

There shall be no notification to the application client that a logical unit has transitioned from one power condition to another. The REQUEST SENSE command (see 6.29) indicates if a logical unit is in the idle power condition or the standby power condition.

Command standards (see 3.1.27) may define for their peripheral device types additional power conditions (e.g., the stopped power condition defined by SBC-3 for direct-access block devices) and extensions to the REQUEST SENSE command for reporting power conditions.

No power condition shall affect the supply of any power required for proper operation of a service delivery subsystem.

Logical units that contain cache memory shall write all cached data to the medium for the logical unit (e.g., as a logical unit would do in response to a SYNCHRONIZE CACHE command as described in SBC-2) prior to entering into any power condition that prevents accessing the media (e.g., before a hard drive stops its spindle motor during transition to the standby power condition).

The power conditions are described in table 52.

**Table 52 — Power Conditions**

Power Condition	Description
active	While in the active power condition (see 3.1.5): a) A device server is capable of responding to all of its supported commands including media access requests; b) A logical unit completes processing of operations in the shortest time when compared to the time required for completion while in the idle or standby power conditions; and c) The SCSI target device may consume more power than when the logical unit is in the idle power condition (e.g., a disk drive's spindle motor may be active).
idle	While in the idle power condition (see 3.1.63): a) A device server is capable of responding to all of its supported commands including media access requests; b) A logical unit may take longer to complete processing a command than it would while in the active power condition (e.g., the device may have to activate some circuitry before processing a command); and c) The power consumed by the SCSI target device should be less than or equal to the power consumed when the logical unit is in the active power condition and may be greater than the power consumed when the logical unit is in the standby power condition.
standby	While in the standby power condition (see 3.1.163): a) A device server is not capable of processing media access commands; and b) The power consumed by the SCSI target device should be less than or equal to the power consumed when the logical unit is in the idle power condition (e.g., a disk drive's spindle motor is stopped).

## 5.10.2 Power condition state machine

### 5.10.2.1 Power condition state machine overview

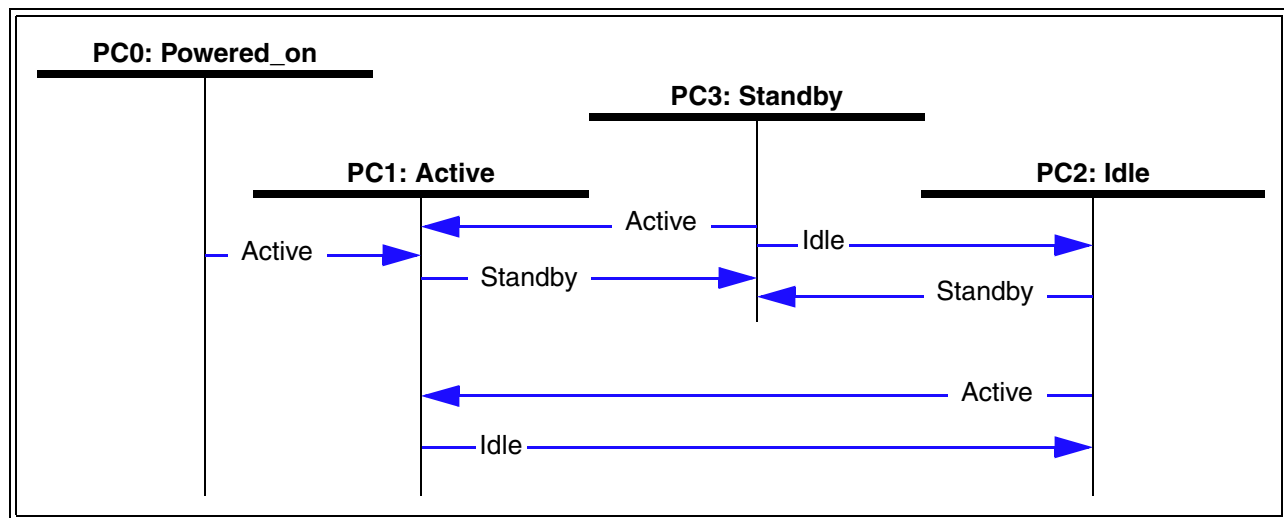
The PC (power condition) state machine describes the logical unit power states and transitions resulting from Power Condition mode page settings.

The PC states are as follows:

- a) PC0:Powered\_on (see 5.10.2.2) (initial state);
- b) PC1:Active (see 5.10.2.3);
- c) PC2:Idle (see 5.10.2.4); and
- d) PC3:Standby (see 5.10.2.5).

The PC state machine shall start in the PC0:Powered\_on state after power on.

Figure 9 describes the PC state machine.



**Figure 9 — Power condition state machine**

#### 5.10.2.2 PC0:Powered\_on state

##### 5.10.2.2.1 PC0:Powered\_on state description

The logical unit shall enter this state upon power on. This state consumes zero time.

##### 5.10.2.2.2 Transition PC0:Powered\_on to PC1:Active

This transition shall occur after the logical unit is ready to begin its power on initialization.

#### 5.10.2.3 PC1:Active state

##### 5.10.2.3.1 PC1:Active state description

While in this state, if power on initialization is not complete, then the logical unit shall complete its power on initialization.

While in this state, if power on initialization is complete, then:

- The logical unit is in the active power condition (see table 52);
- If the idle condition timer is active, then the idle condition timer is running; and
- If the standby condition timer is active, then the standby condition timer is running.

##### 5.10.2.3.2 Transition PC1:Active to PC2:Idle

This transition shall occur after:

- The idle condition timer is active; and
- The idle condition timer is zero.



#### 5.10.2.3.3 Transition PC1:Active to PC3:Standby

This transition shall occur after:

- a) The standby condition timer is active; and
- b) The standby condition timer is zero.

#### 5.10.2.4 PC2:Idle state

##### 5.10.2.4.1 PC2:Idle state description

While in this state:

- a) The logical unit is in the idle power condition (see table 52);
- b) The device server processes the REQUEST SENSE command as described in 6.29; and
- c) If the standby condition timer is active, then the standby condition timer is running.

##### 5.10.2.4.2 Transition PC2:Idle to PC1:Active

This transition shall occur after the device server processes a command that requires the logical unit to be in the PC1:Active state to process the command.

##### 5.10.2.4.3 Transition PC2:Idle to PC3:Standby

This transition shall occur after:

- a) The standby condition timer is active; and
- b) The standby condition timer is zero.

#### 5.10.2.5 PC3:Standby state

##### 5.10.2.5.1 PC3:Standby state description

While in this state:

- a) The logical unit is in the standby power condition (see table 52); and
- b) The device server processes the REQUEST SENSE command as described in 6.29.

##### 5.10.2.5.2 Transition PC3:Standby to PC1:Active

This transition shall occur after the device server processes a command that requires the logical unit to be in the PC1:Active state to process the command.

##### 5.10.2.5.3 PC3:Standby to PC2:Idle

This transition shall occur after the device server processes a command that requires the logical unit to be in the PC2:Idle state to process the command.

## 5.11 Medium auxiliary memory

Some types of media, especially removable media, include a non-volatile memory referred to as MAM (medium auxiliary memory). Medium auxiliary memory is used to store data describing the media and its contents. This standard supports medium auxiliary memory with the READ ATTRIBUTE command (see 6.15) and the WRITE ATTRIBUTE command (see 6.38). These commands are used to retrieve and store information in the medium auxiliary memory in the form of MAM attributes.

A MAM attribute is represented in a format described in 7.3 and is composed of:

- a) An attribute identifier;
- b) An attribute format code;
- c) A bit indicating whether the identified attribute is read only;
- d) An attribute length specifying the number of bytes in the identified attribute value; and
- e) The value of the identified attribute.

There are three types of MAM attributes (see table 53).

**Table 53 — Types of MAM attributes**

Attribute Type	Attribute Source	Example	Readable with READ ATTRIBUTE	Writable with WRITE ATTRIBUTE
Medium	Permanently stored in the medium auxiliary memory during manufacture.	Media Serial Number	Yes	No
Device	Maintained by the device server.	Load Count	Yes	No
Host	Maintained by the application client.	Backup Date	Yes	Yes

Depending on that attribute type, MAM attributes have the states shown in table 54.

**Table 54 — MAM attribute states**

Attribute Type	Attribute State	Description
Medium or Device	Read Only	An application client may read the contents of the MAM attribute with the READ ATTRIBUTE command, but an attempt to clear or change the MAM attribute using the WRITE ATTRIBUTE command shall result in the command being terminated with CHECK CONDITION status. When the READ ONLY bit (see 7.3.1) is one, the attribute is in the read only state.
	Unsupported	The device server does not support the MAM attribute and shall not return it in response to a READ ATTRIBUTE command.
Host	Nonexistent	A host attribute does not exist in the medium auxiliary memory until a WRITE ATTRIBUTE command creates it.
	Read/Write	The MAM attribute has been created using the WRITE ATTRIBUTE command. After the MAM attribute has been created, the contents may be altered using subsequent WRITE ATTRIBUTE commands. A read/write MAM attribute may be returned to the nonexistent state using a WRITE ATTRIBUTE command with the attribute length set to zero. When the READ ONLY bit (see 7.3.1) is zero, the MAM attribute is in the read/write state.

## 5.12 Error history

### 5.12.1 Error history overview

Error history is optional data collected by a logical unit to aid in troubleshooting errors.

The READ BUFFER command (see 6.16.9) provides a method for retrieving error history from the logical unit (see 5.12.2).

The WRITE BUFFER command (see 6.39.14) provides a method for inserting application client error history into the error history (see 5.12.3) and for clearing the error history (see 5.12.4).

The format of the application client error history is defined by the manufacturer of the application client. The format of the error history, including the format of any application client error history included in the error history, is defined by the manufacturer of the logical unit.

### 5.12.2 Retrieving error history with the READ BUFFER command

Device servers may allow the error history to be retrieved using a sequence of READ BUFFER commands on one I\_T nexus.

Error history is returned using error history snapshots. An error history snapshot is the contents of the error history at a specific point in time, created by the device server at vendor specific times or requested by the application client using the READ BUFFER command with certain buffer IDs.

The I\_T nexus being used to retrieve error history snapshot is called the error history I\_T nexus. Only one I\_T nexus is allowed to retrieve an error history snapshot at a time.

To retrieve the complete error history, an application client uses one I\_T nexus to:

- 1) Create an error history snapshot if one does not already exist, establish the I\_T nexus as the error history I\_T nexus, and retrieve the error history directory by sending a READ BUFFER command (see 6.16.9.2) with:
  - A) The MODE field set to 1Ch (i.e., error history);
  - B) The BUFFER ID field set to one of the following:
    - a) If the error history I\_T nexus is expected to be valid:
      - A) 00h (i.e., return error history directory);
      - B) 01h (i.e., return error history directory and create new snapshot);
    - b) If the application client has knowledge obtained by means outside the scope of this standard that the error history I\_T nexus is no longer valid:
      - A) 02h (i.e., return error history directory and establish new error history I\_T nexus); or
      - B) 03h (i.e., return error history directory, establish new error history I\_T nexus, and create new snapshot);
  - C) The BUFFER OFFSET field set to 000000h; and
  - D) The ALLOCATION LENGTH field set to at least 2 088 (i.e., large enough to transfer the complete error history directory);
- 2) Retrieve the error history. The application client uses a data-in buffer size that is a multiple of the offset boundary indicated in the READ BUFFER descriptor (see 6.16.5). For each buffer ID indicated in the error history directory in the range of 10h to EFh, the application client sends one or more READ BUFFER commands (see 6.16.9.3) as follows:
  - 1) Send the first READ BUFFER command with:
    - a) The MODE field set to 1Ch (i.e., error history);
    - b) The BUFFER ID field set to the buffer ID (i.e., an error history data buffer);

- c) The BUFFER OFFSET field set to 000000h; and
- d) The ALLOCATION LENGTH field set to the size of the data-in buffer;
- 2) Until the number of bytes returned by the previous READ BUFFER command does not equal the specified allocation length and/or the total number of bytes returned from the buffer ID equals the maximum available length indicated in the error history directory, send zero or more additional READ BUFFER commands with:
  - a) The MODE field set to 1Ch (i.e., error history);
  - b) The BUFFER ID field set to the buffer ID (i.e., an error history data buffer);
  - c) The BUFFER OFFSET field set to the previous buffer offset plus the previous allocation length; and
  - d) The ALLOCATION LENGTH field set to the size of the data-in buffer;
- and
- 3) Clear the error history I\_T nexus and, depending on the buffer ID, release the error history snapshot by sending a READ BUFFER command with:
  - A) The MODE field set to 1Ch (i.e., error history);
  - B) The BUFFER ID field set to:
    - a) FEh (i.e., clear error history I\_T nexus) (see 6.16.9.4); or
    - b) FFh (i.e., clear error history I\_T nexus and release snapshot) (see 6.16.9.5);
  - C) The BUFFER OFFSET field set to any value allowed by table 187 (see 6.16.9.1) (e.g., 000000h); and
  - D) The ALLOCATION LENGTH field set to any value allowed for the chosen BUFFER ID field value (see 6.16.9.4 or 6.16.9.5) (e.g., 000000h).

While an error history snapshot exists, the device server:

- a) Shall not modify the error history snapshot to reflect any changes to the error history;
- b) May or may not record events that it detects into the error history; and
- c) If it supports the WRITE BUFFER command download application client error history mode (see 6.39.14), shall record any application client error history received into the error history.

The device server shall clear the established error history I\_T nexus and not release the error history snapshot when:

- a) Upon processing of a READ BUFFER command on the error history I\_T nexus with:
  - A) The MODE field set to 1Ch (i.e., error history); and
  - B) The BUFFER ID field set to FEh (i.e., clear error history I\_T nexus) (see 6.16.9.4);
- or
- b) An I\_T nexus loss occurs on the error history I\_T nexus.

The device server shall clear the established error history I\_T nexus and release the error history snapshot when:

- a) Upon processing of a READ BUFFER command using the same I\_T nexus that was used to establish the snapshot with:
  - A) The MODE field set to 1Ch (i.e., error history); and
  - B) The BUFFER ID field set to FFh (i.e., clear error history I\_T nexus and release snapshot) (see 6.16.9.5);
- b) A power on occurs;
- c) A hard reset occurs; or
- d) A logical unit reset occurs.

The device server shall not replace or release the error history snapshot while the error history I\_T nexus is established.

The device server shall implement a vendor specific timer for error history snapshot retrieval. If the vendor specific timer expires:

- a) The device server shall:
  - A) Clear the error history I\_T nexus; and
  - B) Establish a unit attention condition for the error history I\_T nexus with the additional sense code set to ERROR HISTORY I\_T NEXUS CLEARED;
- or
- b) The device server shall:
  - A) Clear the error history I\_T nexus;
  - B) Release the error history snapshot; and
  - C) Establish a unit attention condition for the error history I\_T nexus with the additional sense code set to ERROR HISTORY SNAPSHOT RELEASED.

After an error history snapshot is released, the device server shall resume recording error history for events that it detects.

Error history may also be retrieved by vendor specific methods or other READ BUFFER command sequences that are outside the scope of this standard.

### 5.12.3 Adding application client error history with the WRITE BUFFER command

An application client adds application client detected error history to the error history collected by a logical unit may by sending a WRITE BUFFER command. The application client error history may be recovered as part of the error history (see 5.12.2) or by means outside the scope of this standard and is not used for any logical unit related error recovery.

Error history that contains a mix of application client error history and logical unit error history may be used to correlate an application client-detected error with errors detected internally by the logical unit.

Application clients should minimize the amount of error history they store to prevent error history overflows (see 6.39.14).

### 5.12.4 Clearing error history with the WRITE BUFFER command

An application client clears the portions of the error history that the device server allows to be cleared by sending a WRITE BUFFER command (see 6.39.14) with:

- a) The MODE field set to 1Ch (i.e., download error history);
- b) The BUFFER OFFSET field set to any value allowed by 6.39.14 (e.g., 000000h);
- c) The PARAMETER LIST LENGTH field set to 00001Ah;
- d) In the parameter list, the CLR bit set to one; and
- e) All other fields in the parameter list as 6.39.14 allows.

Clearing error history shall not:

- a) Clear the error history I\_T nexus, if any, if it was created with the READ BUFFER command (see 5.12.2);
- or
- b) Release the error history snapshot, if any, if it was created with the READ BUFFER command (see 5.12.2).

### 5.13 Device clocks

A timestamp may be included in data logged or recorded by a device server. There shall be one timestamp per logical unit.

The timestamp origin shall be one of those specified in table 55.

**Table 55 — TIMESTAMP ORIGIN field**

Code	Description
000b	Timestamp initialized to zero at power-on or as the result of a hard reset
001b	Reserved
010b	Timestamp initialized by the SET TIMESTAMP command (see 6.36)
011b	Timestamp initialized by methods outside the scope of this standard
100b to 111b	Reserved

The Timestamp shall not be affected by an I\_T nexus loss or a logical unit reset.

Once a timestamp is initialized it shall begin counting from that time forward. Once the timestamp is initialized it shall remain in effect until one of the following occurs:

- a) A hard reset occurs;
- b) A SET TIMESTAMP command is processed; or
- c) A method outside the scope of this standard affects the timestamp.

The methods by which a timestamp may be changed are indicated in the Control Extension mode page (see 7.4.7).

If the timestamp is changed by means other than the SET TIMESTAMP command then the device server shall establish a unit attention condition for the initiator port associated with every I\_T nexus (see SAM-4), with the additional sense code set to TIMESTAMP CHANGED.

The TIMESTAMP field format is shown in table 56.

**Table 56 — TIMESTAMP field format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
5	TIMESTAMP							
	(LSB)							

The TIMESTAMP field contains the value established at the last action that set the timestamp incremented by one for every millisecond that has elapsed since the timestamp was set, within vendor specific constraints.

## 5.14 Security Features

### 5.14.1 Security goals and threat model

#### 5.14.1.1 Overview

In many cases, the security goals and threat model used for the Internet are applicable to SCSI commands. The Internet security goals and threat model found in RFC 3552 as they apply to SCSI are summarized in 5.14.1. Terms, concepts, and classes of security techniques that are defined in RFC 3552 are discussed based on their RFC 3552 definitions without modification in this standard.

The security goals and threat model described in 5.14.1 are valid for all SCSI device types. SCSI command standards may modify this model to deal with threats appropriate to specific device types.

#### 5.14.1.2 Security goals

The overall goals of security may be divided into two broad categories:

- a) Communications security (i.e., protecting communications); and
- b) System security.

These goals interact as a result of communications being carried out by systems, with access to those systems provided through communications channels. It is possible to provide security services that independently meet these goals. A common methodology is to secure the communications first and then provide secure access to systems over the secured communication channels.

Communication security is subdivided into the following primary areas of protection:

- a) **Confidentiality:** Preventing unintended entities from seeing the data;
- b) **Cryptographic Data Integrity:** Ensuring that the data that arrives is identical to the data that was sent; and
- c) **Peer Entity Authentication:** Ensuring that the communicating endpoints are the intended peer entities.

Data Origin Authentication (i.e., ensuring that the received data was sent by the authenticated peer) is the combination of Peer Entity Authentication and Cryptographic Data Integrity.

Non Repudiation enhances Data Origin Authentication with the ability to prove to a third party that the sender sent the data that the receiver received.

Cryptographic Data Integrity is called Data Integrity in RFC 3552. The term Cryptographic is added in this standard to distinguish the class of integrity protection required to counter malicious adversaries from the class of integrity protection required to deal with random data corruption (e.g., caused by cosmic rays or electrical noise). Mechanisms used to deal with random data corruption (e.g., parity bits and CRCs) have minimal value against malicious adversaries that are able to modify integrity checks to conceal their modifications to the data. Cryptographic Data Integrity requires knowledge of a secret key in order to modify an integrity check without that modification being detectable. A well designed system provides a high level of assurance that an attacker is unable to learn, guess, discover, or otherwise obtain the required secret key.

In addition to the primary areas, there is another area of control:

- a) **Authorization:** Controlling what an entity is allowed to do. For communications security this is control of the entities with which an entity is allowed to communicate.

Access Control (i.e., controlling what an entity is allowed to access) is a form of Authorization.

Systems security consists of protecting systems from unauthorized usage, inappropriate usage, and denial of service.

#### 5.14.1.3 Threat model

Most secured systems are vulnerable to an attacker equipped with sufficient resources, time, and skills. In order to make designing a security system practical, a threat model is defined to describe the capabilities that an attacker is assumed to be able to deploy (e.g., knowledge, computing capability, and ability to control the system).

The main purposes of a threat model are as follows:

- a) To identify the threats of concern; and
- b) To rule some threats explicitly out of scope.

Most security measures do not provide absolute assurance that an attack has not occurred. Rather, security measures raise the difficulty of accomplishing the attack to well beyond the attacker's assumed capabilities and/or resources. Design of security measures that resist attackers with essentially unlimited capabilities (e.g., certain nation-states) is outside the scope of this standard. Security measures that are susceptible to a level of capability available to some attackers may still be useful for deterring attackers who lack that level of capability, especially when combined with non-technical security measures such as physical access controls.

The computational capability of an attacker is treated as a variable because that capability is inherently a moving target as a result of more powerful processors. The computational capability of an attacker influences design aspects (e.g., key length). Well designed security systems are agile in that they are able to operate not only with different key lengths, but also with different cryptographic algorithms.

The Internet threat model described in RFC 3552 is generally applicable to SCSI, and is specifically applicable when Internet Protocols are used by the SCSI transport (e.g., iSCSI, Fibre Channel via FCIP or iFCP). Its basic assumptions can be summarized as:

- a) End systems engaging in communication are not under the control of the attacker.
- b) The attacker is able to read any communicated IU and undetectably remove, change, or inject forged IUs, including injection of IUs that appear to be from a known and/or trusted system.

Communications security designs are based on an additional assumption that secrets (e.g., keys) used to secure the communications are protected so that an attacker is unable to learn, guess, discover, or otherwise obtain them. A consequence of this assumption is that attacks against secured communications are assumed to begin without with advance knowledge of the secrets used to secure the communications.

#### 5.14.1.4 Types of attacks

The following types of attacks are considered:

- a) Passive attacks (i.e., attacks that only require reading IUs), and
- b) Active attacks (i.e., attacks that require the attacker to change communication and/or engage in communication).



More information on attack types is available in RFC 3552.

Simple passive attacks involve reading communicated data that the attacker was not intended to see (e.g., password, credit card number). More complex passive attacks involve post-processing the communicated data (e.g., checking a challenge-response pair against a dictionary to see if a common word was used as a password).

There are a wide variety of active attacks (e.g., spoofing, replay, insertion, deletion, known plaintext, and modification of communications). Man-in-the-middle attacks are a class of active attacks that involve the attacker inserting itself in the middle of communication, enabling it to intercept all communications without the knowledge of the communicating parties for various purposes (e.g., insertion, deletion, replay, modification and/or inspection via decryption of the communications).

#### **5.14.1.5 SCSI security considerations**

The application of communication security techniques (see RFC 3552) is defined by command standards. This subclause describes specific design considerations in applying the threat model (see 5.14.1.3) to all SCSI device types.

SCSI environments tend not to be fully connected (i.e., there are restrictions on the SCSI device servers with which a SCSI application client is able to communicate) due to the following mechanisms:

- a) Physical and logical connectivity restrictions (e.g., in SCSI to SCSI gateways across different transports);
- b) LUN mapping and masking; and
- c) Transport zoning.

The resulting connectivity is more limited than the Internet security assumption that an off-path attacker is able to transmit to an arbitrary victim (see RFC 3552).

SCSI security designs are also influenced by SCSI being a client-server distributed service model (see SAM-4) that is realized over a number of different SCSI transport protocols and interconnects.

Security functionality may be defined as part of a command set or at the SCSI transport level. Some SCSI transport protocols (e.g., Fibre Channel and iSCSI) define security functionality that provides confidentiality, cryptographic integrity, and peer entity authentication for all communicated data. However, there are situations in which some or all of those mechanisms are not used and there are SCSI communications whose scope spans more than one SCSI Transport Protocol (e.g., via a gateway between iSCSI and FCP). Security that is defined by a command set is appropriate for such situations.

5.14.2 Security associations

5.14.2.1 Principles of SAs

Before an application client and device server begin applying security functions (e.g., data integrity checking, data encryption) to messages (i.e., data that is transferred in either direction between them), they perform a security protocol to create at least one SA (see 5.14.2.3). The result of the SA creation protocol is two sets of SA parameters (see 5.14.2.2), one that is maintained by the application client and one that is maintained by the device server.

In this model, SAs decouple the process of creating a security relationship from its usage in processing security functions. This decoupling allows either the creation or the usage of an SA to be upgraded in response to changing security threats without requiring both processes to be upgraded concurrently.

Figure 10 shows the relationship between application clients and device servers with respect to SAs.

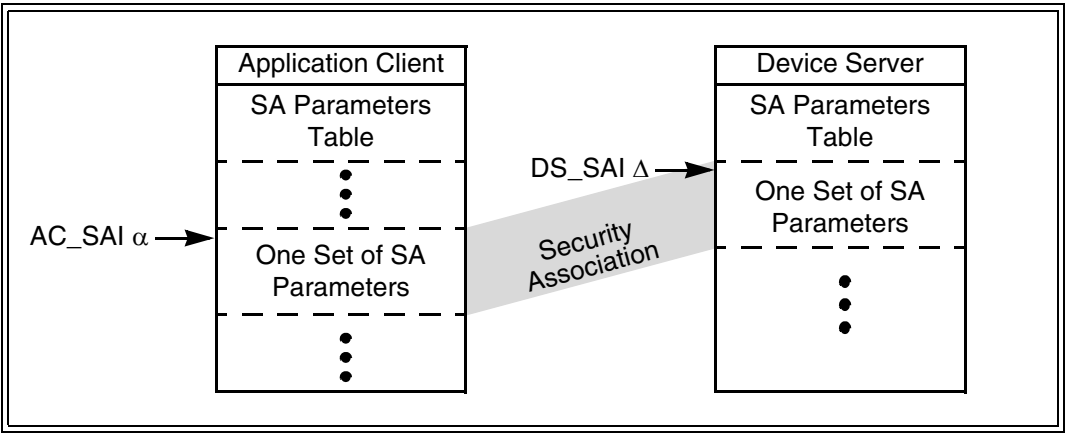


Figure 10 — SA relationships

In both the application client and the device server, the SA parameters are modelled as being stored in an indexed array and the SAI identifies one set of SA parameters within that array. The application client and device server are not required to store the parameters for any given SA in the same array locations. In order to support this implementation flexibility, a single SA is modelled as having two different SAI values (i.e., one for the application client and one for the device server).

The device server shall maintain a single SA parameters table for all I\_T nexuses.

SAs shall not be preserved across a power cycle, hard reset, or logical unit reset. SAs shall not be affected by an I\_T nexus loss.

### 5.14.2.2 SA parameters

Each SAI shall identify at least the SA parameters defined in table 57. Individual security protocols define how the SA parameters are generated and/or used by that security protocol.

**Table 57 — Minimum SA parameters** (part 1 of 3)

Name	Description	Size (bytes) <sup>a</sup>		Scope <sup>b</sup>
		Min.	Max.	
SA parameters that identify and manage the SA.				
AC_SAI	The SAI used by the application client to identify the SA. <sup>c</sup>	4	4	Public
DS_SAI	The SAI used by the device server to identify the SA. <sup>c</sup>	4	4	Public
TIMEOUT	The number of seconds that may elapse after the completion of an SA access operation (i.e., SA creation or SA usage by a command) before the device server should discard the state associated with this SA (e.g., the SA parameters). If SA state is discarded because no SA access operations are received during the specified interval, the device server shall respond to further attempts to access the SA as if the SA had never been created. This parameter shall not be set to zero.	4	4	Public
SA parameters that are incorporated in messages to prevent message replay attacks.				
AC_SQN	A sequence number that is incremented for each response message received by an application client on which a security function is performed and used to detect replay attacks (see 5.14.1.4).	8	8	Public
DS_SQN	A sequence number that is incremented for each request message received by a device server on which a security function is performed and used to detect replay attacks.	8	8	Public
<div><div><sup>a</sup> These size values are guidelines. Specific security protocols may place more exacting size requirements on SA parameters.</div><div><sup>b</sup> Public SA parameters may be transferred outside a SCSI device unencrypted. Secret SA parameters shall not be transferred outside a SCSI device. Fields within a protocol specific SA parameter are Shared or Secret as defined by the applicable SA creation protocol.</div><div><sup>c</sup> SAI values between 0 and 255, inclusive, are reserved.</div><div><sup>d</sup> Nonce SA parameters shall be at least half the size of KEY_SEED SA parameter.</div><div><sup>e</sup> The number of bits of entropy in the KEY_SEED should be as close to the number of bits in the KEY_SEED as possible (see RFC 3766).</div></div>				

**Table 57 — Minimum SA parameters** (part 2 of 3)

Name	Description	Size (bytes) <sup>a</sup>		Scope <sup>b</sup>
		Min.	Max.	
SA parameters that are used by security functions to derive the secret keys that are applied to messages (e.g., for encryption).				
AC_NONCE <sup>d</sup>	A random nonce (see 3.1.115) value that is generated by the application client and used as an input to the key derivation security algorithm specified by the KDF_ID SA parameter during the derivation of an encryption key.	16	64	Public
DS_NONCE <sup>d</sup>	A random nonce value that is generated by the device server and used as an input to the key derivation security algorithm specified by the KDF_ID SA parameter during the derivation of an encryption key.	16	64	Public
KEY_SEED <sup>e</sup>	A value that is known only to the application client and device server that are creating this SA that in combination with the applicable nonce is used to derive the KEYMAT value. The KEY_SEED shall be set to zero as part of completing the SA creation function.	16	64	Secret
KDF_ID	A security algorithm (see 5.14.8) coded value that identifies the KDF used by the application client and device server.	4	4	Public
SA parameters that are used by security functions to secure messages between the application client and device server.				
KEYMAT	A value that is known only to the application client and device server that are participating in this SA that may be subdivided into one or more key values that are used in security functions that secure messages. The contents of KEYMAT depend on the USAGE_TYPE SA parameter value.	14	1 024	Secret
<div><div><sup>a</sup> These size values are guidelines. Specific security protocols may place more exacting size requirements on SA parameters.</div><div><sup>b</sup> Public SA parameters may be transferred outside a SCSI device unencrypted. Secret SA parameters shall not be transferred outside a SCSI device. Fields within a protocol specific SA parameter are Shared or Secret as defined by the applicable SA creation protocol.</div><div><sup>c</sup> SAI values between 0 and 255, inclusive, are reserved.</div><div><sup>d</sup> Nonce SA parameters shall be at least half the size of KEY_SEED SA parameter.</div><div><sup>e</sup> The number of bits of entropy in the KEY_SEED should be as close to the number of bits in the KEY_SEED as possible (see RFC 3766).</div></div>				

**Table 57 — Minimum SA parameters** (part 3 of 3)

Name	Description	Size (bytes) <sup>a</sup>		Scope <sup>b</sup>
		Min.	Max.	
SA parameters that are used by SA management functions.				
USAGE_TYPE	A coded value (see table 58) that indicates the how the SA is used.	2	2	Public
USAGE_DATA	Information associated with how the SA is used (e.g., cryptograph algorithms and key sizes). The contents of USAGE_DATA depend on the USAGE_TYPE SA parameter value.	0	1 024	Public
MGMT_DATA	SA data that is used in ways defined by the SA creation protocol to perform SA management functions (e.g., deletion of the SA).	0	1 024	Protocol specific
<div><sup>a</sup> These size values are guidelines. Specific security protocols may place more exacting size requirements on SA parameters.</div> <div><sup>b</sup> Public SA parameters may be transferred outside a SCSI device unencrypted. Secret SA parameters shall not be transferred outside a SCSI device. Fields within a protocol specific SA parameter are Shared or Secret as defined by the applicable SA creation protocol.</div> <div><sup>c</sup> SAI values between 0 and 255, inclusive, are reserved.</div> <div><sup>d</sup> Nonce SA parameters shall be at least half the size of KEY_SEED SA parameter.</div> <div><sup>e</sup> The number of bits of entropy in the KEY_SEED should be as close to the number of bits in the KEY_SEED as possible (see RFC 3766).</div>				

The USAGE\_TYPE SA parameter shall be one of the values shown in table 58.

**Table 58 — USAGE\_TYPE SA parameter values**

Value <sup>a</sup>	Description	Usage model	Usage data description	Reference
0000h to 0080h	Reserved			
0081h	Tape data encryption	ESP-SCSI <sup>b</sup>	None <sup>c</sup>	SSC-3
0082h to 8000h	Reserved			
8001h	CbCS authentication and credential encryption	ESP-SCSI <sup>b</sup>	None <sup>c</sup>	5.14.6.8
8002h to FFFFh	Reserved			
<sup>a</sup> USAGE_TYPE values between 8000h and CFFFh inclusive place additional constraints on how an SA is to be created as described in 7.6.3.5.13. <sup>b</sup> ESP-SCSI usage is defined in 5.14.7. <sup>c</sup> The usage data length field in the IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.6.3.5.13) shall contain zero.				

### 5.14.2.3 Creating an SA

The SECURITY PROTOCOL IN command (see 6.30) and SECURITY PROTOCOL OUT command (see 6.31) security protocols shown in table 59 are used to create SAs. The process of creating an SA establishes the SA parameter (see 5.14.2.2) values as follows:

- a) Initial values for:
  - A) Both (i.e., application client and device server) sequence numbers set to one;
- b) Unchanging values for the lifetime of the SA:
  - A) Both SAs;
  - B) TIMEOUT;
  - C) KDF\_ID;
  - D) KEYMAT;
  - E) USAGE\_TYPE;
  - F) USAGE\_DATA; and
  - G) MGMT\_DATA;
 and
- c) Values that are zero upon completion of SA creation:
  - A) KEY\_SEED; and
  - B) Both nonces.

**Table 59 — Security protocols that create SAs**

Security Protocol Code	Description	References
40h	SA creation capabilities	7.6.2
41h	IKEv2-SCSI	5.14.4 and 7.6.3

### 5.14.3 Key derivation functions

#### 5.14.3.1 Overview

Table 60 summarizes the key derivation functions defined by this standard.

**Table 60 — Key derivation functions summary**

Security Algorithm Code (see table 88)	Description	Reference
0002 0002h	IKEv2-based iterative HMAC KDF based on SHA-1	5.14.3.3
0002 0005h	IKEv2-based iterative HMAC KDF based on SHA-256	5.14.3.3
0002 0006h	IKEv2-based iterative HMAC KDF based on SHA-384	5.14.3.3
0002 0007h	IKEv2-based iterative HMAC KDF based on SHA-512	5.14.3.3
0002 0004h	IKEv2-based iterative KDF based on AES-128 in XCBC mode	5.14.3.4

### 5.14.3.2 IKEv2-based iterative KDF

In principle, KEYMAT is generated by applying IFUNC to STRING using KEY\_SEED to produce the requested number of KEYMAT bits. In equation notation, the operation is as follows:

$$\text{KEYMAT} = \text{IFUNC}(\text{KEY\_SEED}, \text{STRING})$$

However, accomplishing this may require multiple applications of IFUNC because the number of bits output by IFUNC is not sufficient to meet the number of KEYMAT bits that are to be produced.

The IKEv2-based (see RFC 4306) iterative technique for applying IFUNC is as follows:

- 1) Initialize PREV\_OUTPUT to a null string (i.e., a string that contains no bits);
- 2) Repeat the following function for values of N that increment from one by one to a maximum of 255 or until the total number of bits returned by all invocations of IFUNC equals or exceeds the number of KEYMAT bits that are to be produced, whichever occurs first:  

$$T_N = \text{IFUNC}(\text{KEY\_SEED}, (\text{PREV\_OUTPUT} \parallel \text{STRING} \parallel \text{a byte containing the value } N))$$
 and
- 3) Concatenate the  $T_N$  values (e.g.,  $T_1 \parallel T_2 \parallel T_3$ ) and return as many of the resulting bits as specified by the number of KEYMAT bits that are to be produced input parameter, starting with the first bit in  $T_1$ .

Protocols that call the IFUNC function to generate KEYMAT should ensure that the number of KEYMAT bits requested does not cause N to exceed 255. If N reaches 256, then:

- 1) The requested number of KEYMAT bits is not returned by IFUNC; and
- 2) The request to produce KEYMAT shall be terminated with an error.

### 5.14.3.3 HMAC-based KDFs

If the KDF\_ID is one of those shown in table 61, the key derivation function is a combination of the:

- a) The HMAC function defined in FIPS 198a (see 2.4);
- b) The secure hash function shown in table 61 for the specified KDF\_ID value; and
- c) IKEv2-based iterative KDF technique (see 5.14.3.2).

The technique requires the following inputs from or related to the SA parameters:

- a) AC\_SAI;
- b) DS\_SAI;
- c) AC\_NONCE;
- d) DS\_NONCE;
- e) KEY\_SEED; and
- f) KEYMAT size, in bits.

The USAGE\_TYPE SA parameter and USAGE\_DATA SA parameter (see 5.14.2.2) specify the KEYMAT size as part of the security protocol that performs SA creation (see 5.14.2.3).

The IKEv2-based iterative KDF technique is applied with the following inputs:

- a) IFUNC is the HMAC function defined in FIPS 198a with the translation of inputs names shown in table 61;
- b) KEY\_SEED is the KEY\_SEED SA parameter; and
- c) STRING store the concatenated contents of the following SA parameters:
  - 1) AC\_NONCE;

- 2) DS\_NONCE;
- 3) AC\_SAI; and
- 4) DS\_SAI.

**Table 61 — HMAC-based key derivation functions**

FIPS 198a inputs selected by KDF_ID	KDF_ID (see table 60)			
	0002 0002h	0002 0005h	0002 0006h	0002 0007h
<i>H</i> (i.e., hash function)	SHA-1 (see table 62)	SHA-256 (see table 62)	SHA-384 (see table 62)	SHA-512 (see table 62)
<i>B</i> (i.e., hash input block size) <sup>a</sup>	64	64	128	128
<i>L</i> (i.e., hash output block size) <sup>a</sup>	20	32	48	64
<i>t</i> (i.e., MAC size) <sup>a</sup>				
<i>K</i> (i.e., key)	KEY_SEED SA parameter			
<i>text</i>	STRING as defined in this subclause and used in 5.14.3.2			
<sup>a</sup> In accordance with FIPS 198a, all sizes are shown in bytes.				

Details of the hash functions that act as inputs to the FIPS 198a HMAC function are shown in table 62.

**Table 62 — Hash functions used by HMAC based on KDF\_ID**

KDF_ID (see table 60)	Function	Description
0002 0002h	SHA-1	HMAC input <i>H</i> is the SHA-1 secure hash function defined in FIPS 180-2 with Change Notice 1 (see 2.4).
0002 0005h	SHA-256	HMAC input <i>H</i> is the SHA-256 secure hash function defined in FIPS 180-2 with Change Notice 1.
0002 0006h	SHA-384	HMAC input <i>H</i> is the SHA-384 secure hash function defined in FIPS 180-2 with Change Notice 1.
0002 0007h	SHA-512	HMAC input <i>H</i> is the SHA-512 secure hash function defined in FIPS 180-2 with Change Notice 1.

#### 5.14.3.4 AES-XCBC-PRF-128 IKEv2-based iterative KDF

If the KDF\_ID is 0002 0004h, the key derivation function is a combination of the:

- a) AES-XCBC-PRF-128 secure hash function defined in RFC 4434 (see 2.5) and RFC 3566; and
- b) IKEv2-based iterative KDF technique (see 5.14.3.2).

The technique requires the following inputs from or related to the SA parameters:

- a) AC\_SAI;
- b) DS\_SAI;
- c) AC\_NONCE;



- d) DS\_NONCE;
- e) KEY\_SEED; and
- f) The number of KEYMAT bits that are to be produced.

The IKEv2-based iterative KDF technique is applied with the following inputs:

- a) IFUNC is the AES-XCBC-PRF-128 secure hash function with the translation of inputs names shown in table 63;
- b) KEY\_SEED is the KEY\_SEED SA parameter; and
- c) STRING store the concatenated contents of the following SA parameters:
  - 1) 1) AC\_NONCE;
  - 2) 2) DS\_NONCE;
  - 3) 3) AC\_SAI; and
  - 4) 4) DS\_SAI.

**Table 63 — RFC 3566 parameter translations KDF based on AES-XCBC-PRF-128**

RFC 3566 Parameter	Translation
K (i.e., key)	KEY_SEED SA parameter
M (i.e., message)	STRING as defined in this subclause and used in 5.14.3.2

#### 5.14.4 Using IKEv2-SCSI to create a security association

##### 5.14.4.1 Overview

The IKEv2-SCSI protocol is a subset of the IKEv2 protocol (see RFC 4306) that this standard defines for use in the creation and maintenance of an SA (see 3.1.148).

An IKEv2-SCSI SA creation transaction (see 3.1.123) shall only be initiated by the application client.

The IKEv2-SCSI protocol creates the following pair of IKE SAs (see RFC 4306):

- a) An SA that protects data sent from the application client to the device server; and
- b) An SA that protects data sent from the device server to the application client.

An IKEv2-SCSI SA creation transaction consists of the following steps:

- 1) Device Server Capabilities step (see 5.14.4.7): The application client determines the device server's cryptographic capabilities;
- 2) Key Exchange step (see 5.14.4.8): The application client and device server:
  - A) Perform a key exchange;
  - B) Determine SAs (see 3.1.149);
  - C) Generate the shared keys used for SA management (e.g., SA creation and deletion) (see 5.14.4.10); and
  - D) May complete the generation of the SA (see 5.14.4.11); and
- 3) Authentication step (see 5.14.4.9): Unless omitted by application client and device server negotiations in the previous steps:
  - A) The application client and device server authenticate:
    - a) Each other;
    - b) The key exchange; and
    - c) The capability selection;

- and
- B) Complete the generation of the SA (see 5.14.4.11).

The values in the SECURITY PROTOCOL field and the SECURITY PROTOCOL SPECIFIC field in the SECURITY PROTOCOL IN command (see 6.30) and SECURITY PROTOCOL OUT command (see 6.31) identify the step of the IKEv2-SCSI protocol (see 7.6.3.2).

The Key Exchange step and the Authentication step depend on the results from the Device Capabilities step in order to create an SA. During the IKEv2-SCSI Key Exchange step, the application client and device server perform independent computations to construct the following sets of shared keys:

- a) Shared keys that are used by the IKEv2-SCSI Authentication step;
- b) Shared keys that are used by the IKEv2-SCSI Authentication step and to delete the SA; and
- c) Shared keys are used by SCSI the usage type specific operations that obtain security from the generated SA.

More details about these shared keys are provided in 5.14.4.4.

An application client may or may not:

- a) Proceed to the Key Exchange step after the Device Server Capabilities step; or
- b) Perform a separate Device Server Capabilities step for each IKEv2-SCSI SA creation transaction.

If the device server's capabilities have changed, the Authentication step returns an error, and the Key Exchange step may return an error.

Changes in the device server's capabilities do not take effect until at least one application client has been notified of the new capabilities via the parameter data returned by the Device Server Capabilities step.

After a Device Capabilities step, the application client performs SA creation by sending a sequence of two or four IKEv2-SCSI commands over a single I\_T\_L nexus to the device server. The following commands constitute an IKEv2-SCSI CCS (see 3.1.66):

- a) If the Authentication step is skipped (see 5.14.4.6):
  - 1) A Key Exchange step SECURITY PROTOCOL OUT command (see 5.14.4.8.2); and
  - 2) A Key Exchange step SECURITY PROTOCOL IN command (see 5.14.4.8.3);
 or
- b) If the Authentication step is performed:
  - 1) A Key Exchange step SECURITY PROTOCOL OUT command (see 5.14.4.8.2);
  - 2) A Key Exchange step SECURITY PROTOCOL IN command (see 5.14.4.8.3);
  - 3) An Authentication step SECURITY PROTOCOL OUT command (see 5.14.4.9.2); and
  - 4) An Authentication step SECURITY PROTOCOL IN command (see 5.14.4.9.3).

The device server shall process each command in the IKEv2-SCSI CCS to completion before returning status. While a command in the IKEv2-SCSI CCS is being processed by the device server, the application client may use the REQUEST SENSE command (see 5.14.5) to ascertain the device server's progress for the command.

When an error is encountered, the device server or application client may abandon the IKEv2-SCSI CCS before the SA is created (see 5.14.4.12).

The device server shall maintain state for the IKEv2-SCSI CCS on a given I\_T\_L nexus from the time the Key Exchange step SECURITY PROTOCOL OUT command is completed with GOOD status until one of the following occurs:

- a) The IKEv2-SCSI CCS completes successfully;
- b) The IKEv2-SCSI CCS is abandoned as described in 5.14.4.12;
- c) The SA being created by the IKEv2-SCSI CCS is deleted as described in 5.14.4.13;
- d) The number of seconds specified in the IKEV2-SCSI PROTOCOL TIMEOUT field of the IKEv2-SCSI Timeout Values payload (see 7.6.3.5.14) in the Key Exchange step SECURITY PROTOCOL OUT parameter data elapses and none of the following commands have been received:
  - A) The next command in the IKEv2-SCSI CCS; or
  - B) A REQUEST SENSE command;
 or
- e) One of the following event-related SCSI device conditions (see SAM-4) occurs:
  - A) Power cycle;
  - B) Hard reset;
  - C) Logical unit reset; or
  - D) I\_T nexus loss.

If the device server receives a SECURITY PROTOCOL OUT or SECURITY PROTOCOL IN command with the SECURITY PROTOCOL field set to IKEv2-SCSI (i.e., 41h) on an I\_T\_L nexus other than the one for which IKEv2-SCSI CCS state is being maintained, then:

- a) An additional IKEv2-SCSI CCS may be started; or
- b) The command may be terminated with CHECK CONDITION status, with the sense key set to ABORTED COMMAND, and the additional sense code set to CONFLICTING SA CREATION REQUEST.

Except for the PERSISTENT RESERVE OUT command (see 5.7.4) and the cases described in this subclause, a device server that is maintaining a IKEv2-SCSI CCS state on a particular I\_T\_L nexus shall not alter its processing of new commands received on that I\_T\_L nexus.

If all of the following conditions are true:

- a) The device server includes the following algorithm descriptors in the IKEv2-SCSI SA Creation Capabilities payload (see 7.6.3.5.11) in the parameter data returned by the Device Server Capabilities step SECURITY PROTOCOL IN command (see 5.14.4.7):
  - A) An SA\_AUTH\_OUT algorithm descriptor (see 7.6.3.6.6) with the ALGORITHM IDENTIFIER field set to SA\_AUTH\_NONE; and
  - B) An SA\_AUTH\_IN algorithm descriptor (see 7.6.3.6.6) with the ALGORITHM IDENTIFIER field set to SA\_AUTH\_NONE;
 and
- b) The application client sends the following algorithm descriptors to the device server in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.6.3.5.12) in the Key Exchange step SECURITY PROTOCOL OUT command (see 5.14.4.8.2):
  - A) An SA\_AUTH\_OUT algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA\_AUTH\_NONE; and
  - B) An SA\_AUTH\_IN algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA\_AUTH\_NONE;

then:

- a) The Authentication step is skipped;
- b) The IKEv2-SCSI CCS consists of the two Key Exchange step commands;
- c) The device server requires the IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.6.3.5.13) to be present in the parameter data sent by the Key Exchange step SECURITY PROTOCOL OUT command;

- d) The device server returns the IKEv2-SCSI SAUT Cryptographic Algorithms payload in the parameter data returned by the Key Exchange step SECURITY PROTOCOL IN command; and
- e) SA creation occurs upon the completion of the Key Exchange step.

Proper operation of an IKEv2-SCSI CCS depends on SA\_AUTH\_NONE being used in both the Authentication step SECURITY PROTOCOL OUT command and the Authentication step SECURITY PROTOCOL IN command, or SA\_AUTH\_NONE not being used by either command. Processing requirements placed on the Key Exchange step SECURITY PROTOCOL OUT command (see 7.6.3.6.6) ensure that this dependency is satisfied.

If no other errors are detected and any of the following conditions are true:

- a) The device server does not include an SA\_AUTH\_OUT algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA\_AUTH\_NONE in the IKEv2-SCSI SA Creation Capabilities payload in the parameter data returned by the Device Server Capabilities step SECURITY PROTOCOL IN command;
- b) The device server does not include an SA\_AUTH\_IN algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA\_AUTH\_NONE in the IKEv2-SCSI SA Creation Capabilities payload in the parameter data returned by the Device Server Capabilities step SECURITY PROTOCOL IN command; or
- c) The application client does not set the ALGORITHM IDENTIFIER field to SA\_AUTH\_NONE in the SA\_AUTH\_OUT algorithm descriptor and the SA\_AUTH\_IN algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload sent to the device server in the Key Exchange step SECURITY PROTOCOL OUT command;

then:

- a) The Authentication step is processed;
- b) The IKEv2-SCSI CCS consists of the two Key Exchange step commands and two Authentication step commands;
- c) The device server requires the IKEv2-SCSI SAUT Cryptographic Algorithms payload to be absent from the parameter data sent by the Key Exchange step SECURITY PROTOCOL OUT command;
- d) The device server omits the IKEv2-SCSI SAUT Cryptographic Algorithms payload from the parameter data returned by the Key Exchange step SECURITY PROTOCOL IN command;
- e) The device server requires the IKEv2-SCSI SAUT Cryptographic Algorithms payload to be present in the parameter data sent by the Authentication step SECURITY PROTOCOL OUT command;
- f) The device server returns the IKEv2-SCSI SAUT Cryptographic Algorithms payload in the parameter data returned by the Authentication step SECURITY PROTOCOL IN command; and
- g) SA creation occurs upon the completion of the Authentication step.

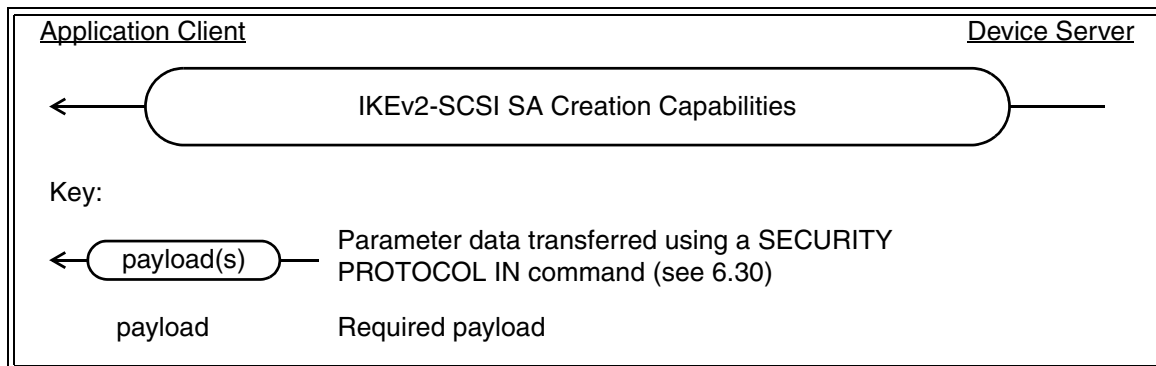
SA participants should perform the Authentication step unless man-in-the-middle attacks (see 5.13.1.4) are not of concern or are prevented by a means outside the scope of this standard (e.g., physical security of the transport).

NOTE 11 - Omission of the Authentication step provides no defense against a man-in-the-middle adversary that is capable of modifying SCSI commands. Such an adversary is able to insert itself as an intermediary on the created SA without knowledge of the SA participants, thereby completely subverting the intended security. Omission of the Authentication step is only appropriate in environments where the absence of such adversaries is assured by other means (e.g., a direct physical connection between the systems on which the application client and device server or use of end-to-end security in the SCSI transport security such as FC-SP).

#### 5.14.4.2 IKEv2-SCSI Protocol summary

This subclause summarizes the IKE-v2-SCSI payloads (see 7.6.3.5) that are exchanged between an application client and a device server during all steps of an IKEv2-SCSI SA creation transaction (see 3.1.123) using message diagrams. Each IKEv2-SCSI step (see 5.14.4.1) is shown in a separate figure. The contents of a payload (e.g., Key Exchange) may not be the same in both directions of transfer.

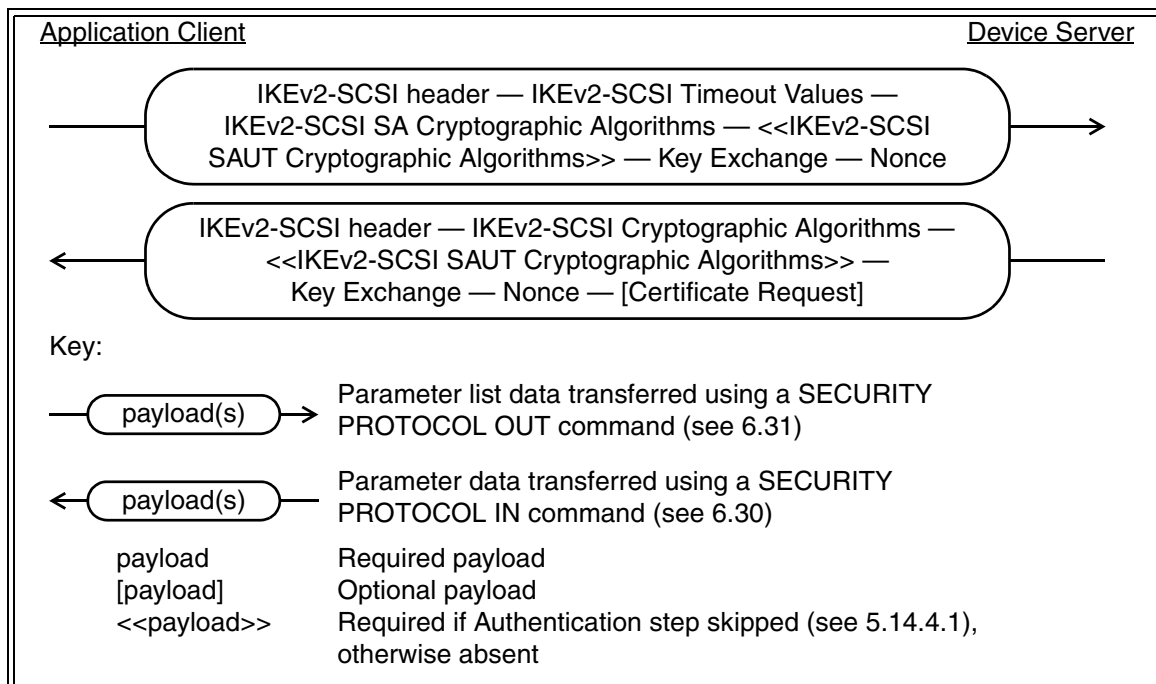
Figure 8 shows the Device Server Capabilities step (see 5.14.4.7). The Device Server Capabilities step consists of a SECURITY PROTOCOL IN command carrying an IKEv2-SCSI SA Creation Capabilities payload (see 7.6.3.5.11). The IKEv2-SCSI header is not used.



**Figure 11 — IKEv2-SCSI Device Server Capabilities step**

The IKEv2-SCSI SA Creation Capabilities payload indicates the device server's capabilities for SA creation.

Figure 12 shows the Key Exchange step (see 5.14.4.8). The Key Exchange step consists of a SECURITY PROTOCOL OUT command followed by a SECURITY PROTOCOL IN command.



**Figure 12 — IKEv2-SCSI Key Exchange step**

The IKEv2-SCSI Timeout Values payload (see 7.6.3.5.14) contains timeouts for SA creation and usage.

The IKEv2-SCSI SA Cryptographic Algorithms payloads (see 7.6.3.5.12) are used to select and agree on the cryptographic algorithms used for creating the SA.

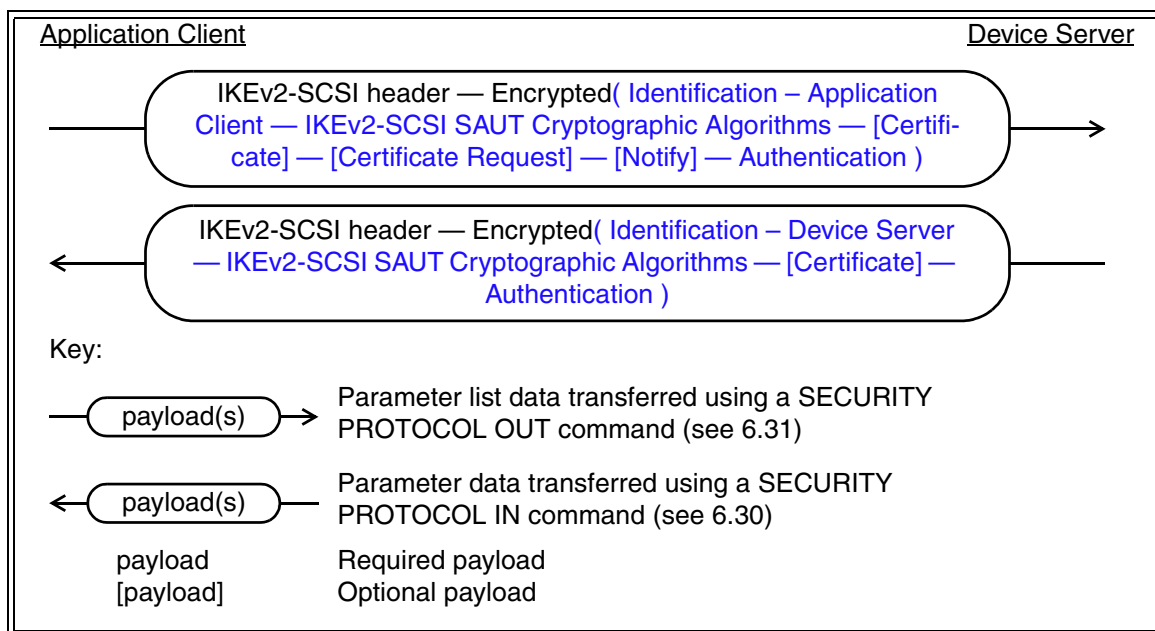
If the Authentication step is skipped (see 5.14.4.1), the IKEv2-SCSI SAUT Cryptographic Algorithms payloads (see 7.6.3.5.13) are used to select and agree on usage of the SA and the cryptographic algorithms used by the created

SA. If the Authentication step is processed (i.e., not skipped), the SA usage and algorithms selection is performed there.

The Key Exchange payload (see 7.6.3.5.3) and Nonce payload (see 7.6.3.5.7) are part of the key and nonce exchanges that are used to generate the IKEv2-SCSI keys (see 3.1.65) and SA keys (see 3.1.125).

The optional Certificate Request payload or payloads (see 7.6.3.5.5) enables the device server to request a certificate from the application client. If the Authentication step is being skipped (see 5.14.4.1), the device server shall not include any Certificate Request payloads in the parameter data. Use of the Certificate Request payload is described in 5.14.4.3.

Figure 13 shows the Authentication step (see 5.14.4.9). The Authentication step consists of a SECURITY PROTOCOL OUT command followed by a SECURITY PROTOCOL IN command.



**Figure 13 — IKEv2-SCSI Authentication step**

An Encrypted payload (see 7.6.3.5.10) contains all other Authentication step payloads that are protected using the cryptographic algorithms determined by the IKEv2-SCSI SA Cryptographic Algorithms payloads (see 7.6.3.5.12) in the Key Exchange step.

The Identification payloads (see 7.6.3.5.4) contain the identities to be authenticated. These identities are not required to be SCSI names or identifiers.

The IKEv2-SCSI SAUT Cryptographic Algorithms payloads (see 7.6.3.5.13) are used to select and agree on usage of the SA and the cryptographic algorithms used by the created SA.

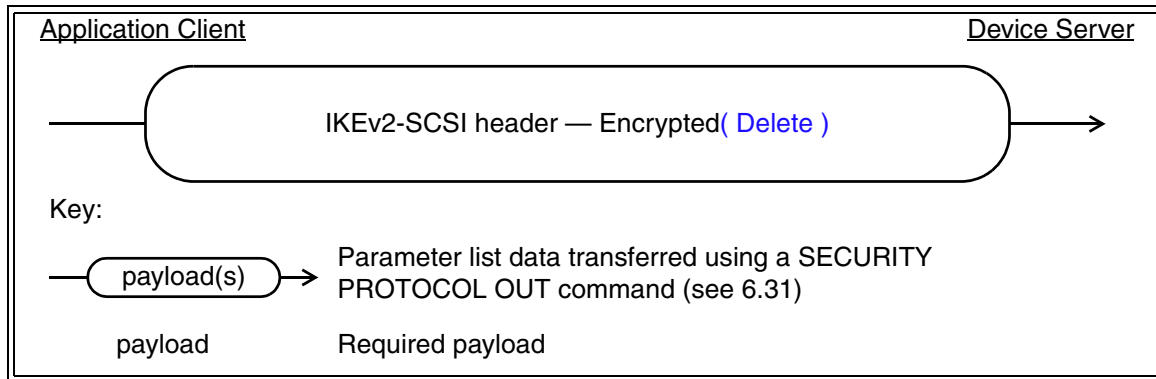
The optional Certificate payload or payloads (see 7.6.3.5.5) respond to the Certificate Request payload(s) sent by the device server in the Key Exchange step SECURITY PROTOCOL IN command.

The optional Certificate Request payload or payloads (see 7.6.3.5.5) allows an application client to request the delivery of a Certificate payload (see 7.6.3.5.5) in the parameter data for the Authentication step SECURITY PROTOCOL IN command (see 5.14.4.3).

The optional Notify payload (see 7.6.3.5.8) provides a means for the application client to inform the device server that this is the only SA being used between them, and that the device server should discard state for any other SAs created by the same application client.

The Authenticate payloads (see 7.6.3.5.6) authenticate not only the SA participants, but also the entire protocol sequence (e.g., the Authenticate payloads prevent a man-in-the-middle attack from succeeding).

Figure 14 shows the Delete operation (see 5.14.4.13). The Delete operation consists of a SECURITY PROTOCOL OUT command.



**Figure 14 — IKEv2-SCSI Delete operation**

An Encrypted payload (see 7.6.3.5.10) contains the Delete payload that is protected using the cryptographic algorithms determined by the IKEv2-SCSI SA Cryptographic Algorithms payloads (see 7.6.3.5.12) in the Key Exchange step that was used to create the SA.

The Delete payload (see 7.6.3.5.9) specifies the SA to be deleted.

#### 5.14.4.3 Handling of the Certificate Request payload and the Certificate payload

As detailed in this subclause, a Certificate Request payload (see 7.6.3.5.5) in one set of parameter data requests the delivery of a Certificate payload (see 7.6.3.5.5) in the next set of parameter data transferred. The purpose of these IKEv2-SCSI protocol elements is as follows:

- a) Each SA participant is allowed to require the delivery of a Certificate payload by the other SA participant for use in authentication; and
- b) Each Certificate Request payload indicates the trust anchors (see RFC 4306) list used by the device server or application client when PKI-based Authentication is being used with certificates that are not self signed (see RFC 3280).

The presence of one or more Certificate Request payloads in the Key Exchange step SECURITY PROTOCOL IN command (see 5.14.4.8.3) parameter data indicates that the device server requires the application client to send a Certificate payload in the Authentication step SECURITY PROTOCOL OUT command (see 5.14.4.9.2).

The presence of one or more Certificate Request payloads in the Authentication step SECURITY PROTOCOL OUT command parameter list specifies that the application client requires the device server to send a Certificate payload in the Authentication step SECURITY PROTOCOL IN command (see 5.14.4.9.3).

If any Certificate payloads are included in the parameter data, the first Certificate payload shall contain the public key used to verify the Authentication payload. Additional Certificate payloads may be sent to assist in establishing a chain of trust from the certificate in the first payload to a trust anchor.

The application client and device server may use different authentication methods that require or do not require the use of Certificate payloads, and the presence or absence of Certificate Request payloads and Certificate payloads may vary in any of the commands described in this subclause.

#### 5.14.4.4 Summary of IKEv2-SCSI shared keys nomenclature and shared key sizes

To facilitate use of the normative references made by this standard, the IKEv2-SCSI shared keys (see 3.1.155) are named as shown in table 64.

**Table 64 — IKEv2-SCSI shared key names and SA shared key names**

Name	Description	SA parameter (see 3.1.126) that stores this shared key
Shared keys used only during Authentication step		
SK_pi	The shared key used to construct the Authentication payload (see 7.6.3.5.6) for the SECURITY PROTOCOL OUT parameter list in the Authentication step (see 5.14.4.9.2).	shall not be stored in any SA parameter
SK_pr	The shared key used to construct the Authentication payload for the SECURITY PROTOCOL IN parameter data in the Authentication step.	
Shared keys used during IKEv2-SCSI SA creation and management		
SK_ai <sup>a</sup>	The shared key used to integrity check the Encrypted payload in the SECURITY PROTOCOL OUT parameter list in the: a) Authentication step; and b) IKEv2-SCSI Delete operation (see 5.14.4.13).	MGMT_DATA
SK_ar <sup>a</sup>	The shared key used to integrity check the Encrypted payload (see 7.6.3.5.10) in the SECURITY PROTOCOL IN parameter data in the Authentication step (see 5.14.4.9.3).	
SK_ei <sup>a</sup>	The shared key used to encrypt the Encrypted payload in the SECURITY PROTOCOL OUT parameter list in the: a) Authentication step; and b) IKEv2-SCSI Delete operation.	
SK_er <sup>a</sup>	The shared key used to encrypt the Encrypted payload in the SECURITY PROTOCOL IN parameter data in the Authentication step.	
Shared key used to construct the SA keys (see 3.1.125)		
SK_d	The shared key material that is used as input to the KDF that generates the KEYMAT SA parameter (see 3.1.126) bytes for the SA.	KEY_SEED
<sup>a</sup> The SA keys (see 3.1.125) SK_ai, SK_ar, SK_ei, and SK_er are stored in the KEYMAT SA parameter. SK_ai is intended to be used for integrity checking data in a Data-Out Buffer. SK_ar is intended to be used for integrity checking data in a Data-In Buffer. SK_ei is intended to be used for encrypting data in a Data-Out Buffer. SK_er is intended to be used for encrypting data in a Data-In Buffer.		



The sizes of the shared keys are determined as shown in table 65.

**Table 65 — Shared key size determination**

Name	Shared key size determination	
	Separate encryption and integrity checking <sup>a, b</sup>	Combined mode encryption and integrity checking <sup>a, b</sup>
SK_ai and SK_ar <sup>c</sup>	The shared key length shown in table 427 for the value in the ALGORITHM IDENTIFIER field of the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.4) <sup>b</sup>	0
SK_er and SK_ei <sup>c</sup>	The shared key length shown in the KEY LENGTH field of the ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.2) <sup>b</sup>	The shared key length shown in the KEY LENGTH field of the ENCR IKEv2-SCSI cryptographic algorithm descriptor plus the number of salt bytes shown in table 423 (see 7.6.3.6.2) <sup>b</sup>
SK_pi and SK_pr <sup>c</sup>	The shared key length is equal to the PRF output length (see table 425) associated with the value in the ALGORITHM IDENTIFIER field of the PRF IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.3) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.6.3.5.12)	
SK_d		
<sup>a</sup> The use of combined mode encryption and integrity checking is indicated by the AUTH_COMBINED value in the algorithm identifier field of the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.4).		
<sup>b</sup> For the shared keys used to create an SA, the algorithm descriptor is located in an IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.6.3.5.12). For the shared keys used after the SA is created (i.e., the KEYMAT SA parameter), the algorithm descriptor is located in an IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.6.3.5.13).		
<sup>c</sup> To accommodate the need for two shared keys of the specified size, the shared key length shown is doubled for the purposes of shared key generation.		

#### 5.14.4.5 IKEv2-SCSI usage of pre-shared keys

If the Authentication payload (see 7.6.3.5.6) AUTHENTICATION DATA field contents are computed using pre-shared keys (e.g., if the applicable algorithm identifier is 00F9 0002h, Shared Key Message Integrity Code), then the following requirements apply in addition to those found in RFC 4306:

- a) A pre-shared key shall be associated with one identity;
- b) The same pre-shared key shall not be used to authenticate both an application client and a device server;
- c) Use of the same pre-shared key for a group of application clients or a group of device servers is strongly discouraged, because it enables any member of the group to impersonate any other member;
- d) The means for provisioning pre-shared keys are outside the scope of this standard;
- e) The pre-shared keys may be provisioned as follows:
  - A) At the time of manufacturing;
  - B) During device or system initialization; or
  - C) Any time thereafter;
- f) The following requirements from RFC 4306 apply to the interfaces for provisioning pre-shared keys:
  - A) ASCII strings of at least 64 bytes shall be supported;
  - B) A null terminator shall not be added to any input before it is used as a pre-shared key;
  - C) A hexadecimal ASCII encoding of the pre-shared key shall be supported; and

- D) ASCII encodings other than hexadecimal may be supported. Support for any such encoding shall include specification of the algorithm for translating the encoding to a binary string as part of the interface;
- and
- g) Information about the size of the pre-shared key shall be stored at the same time that the pre-shared key is stored.

#### 5.14.4.6 Constraints on skipping the Authentication step

In the Device Server Capabilities step (see 5.14.4.7), the parameter data returned by the SECURITY PROTOCOL IN command (see 7.6.2.3.2) contains the IKEv2-SCSI SA Creation Algorithms payload (see 7.6.3.5.11) that contains one or more SA\_AUTH\_OUT IKEv2-SCSI cryptographic algorithm descriptors (see 7.6.3.6.6) and one or more SA\_AUTH\_OUT IKEv2-SCSI cryptographic algorithm descriptors.

The device server permits the Authentication step to be omitted (see 5.14.4.1) if all of the following are true:

- a) The ALGORITHM IDENTIFIER field is set to SA\_AUTH\_NONE (see 7.6.3.6.6) in one of the SA\_AUTH\_OUT IKEv2-SCSI cryptographic algorithm descriptors returned in the Device Server Capabilities step; and
- b) The ALGORITHM IDENTIFIER field is set to SA\_AUTH\_NONE in one of the SA\_AUTH\_IN IKEv2-SCSI cryptographic algorithm descriptors returned in the Device Server Capabilities step.

The methods for configuring a device server to return SA\_AUTH\_NONE are outside the scope of this standard. Device servers shall not be manufactured to return SA\_AUTH\_NONE as an Authentication payload authentication algorithm type in the Device Server Capabilities step.

In the Key Exchange step SECURITY PROTOCOL OUT command (see 5.14.4.8.2), the application client requests that the Authentication step be omitted by setting the ALGORITHM IDENTIFIER field to SA\_AUTH\_NONE in the SA\_AUTH\_OUT cryptographic algorithm descriptor and in the SA\_AUTH\_IN cryptographic algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.6.3.5.12).

To ensure adequate SA security, the application client should not select the SA\_AUTH\_NONE value as an Authentication payload authentication algorithm type unless:

- a) An SA\_AUTH\_OUT IKEv2-SCSI cryptographic algorithm descriptor and an SA\_AUTH\_IN IKEv2-SCSI cryptographic algorithm descriptor from the Device Server Capabilities step indicates SA\_AUTH\_NONE availability; and
- b) The application client is configured to omit the Authentication step.

NOTE 12 - When SA\_AUTH\_NONE is used, IKEv2-SCSI has no protection against any man-in-the-middle attacks. Enabling return of the SA\_AUTH\_NONE authentication algorithm type in the Device Capabilities step, and allowing an application client to select SA\_AUTH\_NONE in the Key Exchange step are administrative security policy decisions that absence of authentication is acceptable, and should only be made with a full understanding of the security consequences of the lack of authentication. Such decisions should only be made in situations where active attacks on IKEv2-SCSI are not of concern (e.g., direct attachment of a SCSI initiator device and a SCSI target device, or an end-to-end secure service delivery subsystem such as FCP over Fibre Channel secured by an end-to-end FC-SP SA).

#### 5.14.4.7 Device Server Capabilities step

In the Device Server Capabilities step, the application client sends a SECURITY PROTOCOL IN command (see 6.30) with the SECURITY PROTOCOL field set to SA creation capabilities (i.e., 40h) and the SECURITY PROTOCOL SPECIFIC field set to 0101h.

The device server returns the SECURITY PROTOCOL IN parameter data specified by the SECURITY PROTOCOL SPECIFIC field (see 7.6.2.2) and the parameter data (see 7.6.2.3.2) contains an IKEv2-SCSI SA Creation Capabilities payload (see 7.6.3.5.11).

In the Device Server Capabilities step, the device server shall return parameter data containing the IKEv2-SCSI cryptographic algorithm descriptors (see 7.7.3.6) in at least one complete row shown in table 66.

**Table 66 — Device Server Capabilities step parameter data requirements**

<b>IKEv2-SCSI cryptographic algorithm descriptor <sup>a</sup></b>					
<b>ENCR</b> (see 7.6.3.6.2)	<b>PRF</b> (see 7.6.3.6.3)	<b>INTEG</b> (see 7.6.3.6.4)	<b>D-H</b> (see 7.6.3.6.5)	<b>SA_AUTH_OUT</b> (see 7.6.3.6.6)	<b>SA_AUTH_IN</b> (see 7.6.3.6.6)
The ALGORITHM IDENTIFIER field set to 8001 0014h (i.e., AES-GCM) and the KEY LENGTH field set to 0010h	The ALGORITHM IDENTIFIER field set to 8002 0005h (i.e., IKEv2-use based on SHA-256)	The ALGORITHM IDENTIFIER field set to F003 0001h (i.e., AUTH_COMBINED)	The ALGORITHM IDENTIFIER field set to 8004 000Eh (i.e., 2 048-bit MODP group (finite field D-H))	The ALGORITHM IDENTIFIER field set to 00F9 0001h (i.e., RSA Digital Signature)	The ALGORITHM IDENTIFIER field set to 00F9 0001h (i.e., RSA Digital Signature)
The ALGORITHM IDENTIFIER field set to 8001 000Ch (i.e., AES-CBC) and the KEY LENGTH field set to 0020h	The ALGORITHM IDENTIFIER field set to 8002 0007h (i.e., IKEv2-use based on SHA-512)	The ALGORITHM IDENTIFIER field set to 8003 000Eh (i.e., AUTH_HMAC_SHA2_512_256)	The ALGORITHM IDENTIFIER field set to 8004 0015h (i.e., 521-bit random ECP group)	The ALGORITHM IDENTIFIER field set to 00F9 000Bh (i.e., ECDSA with SHA-512 on the P-521 curve)	The ALGORITHM IDENTIFIER field set to 00F9 000Bh (i.e., ECDSA with SHA-512 on the P-521 curve)
<sup>a</sup> The IKEv2-SCSI cryptographic algorithm descriptors shown in all the columns in at least one row in this table shall be returned in the Device Server Capabilities step parameter data.					

In the Device Server Capabilities step, the device server shall return parameter data containing one SA\_AUTH\_OUT IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field is set to SA\_AUTH\_NONE and one SA\_AUTH\_IN IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field is set to SA\_AUTH\_NONE if any of the following are true:

- The ALGORITHM IDENTIFIER field is set to SA\_AUTH\_NONE in one of the SA\_AUTH\_OUT IKEv2-SCSI cryptographic algorithm descriptors returned in the Device Server Capabilities step; or
- The ALGORITHM IDENTIFIER field is set to SA\_AUTH\_NONE in one of the SA\_AUTH\_IN IKEv2-SCSI cryptographic algorithm descriptors returned in the Device Server Capabilities step.

The device server capabilities returned in the SECURITY PROTOCOL IN parameter data may be changed at any time via interfaces that are outside the scope of this standard, however, such changes shall not take effect until at least one application client has been notified of the new capabilities via the parameter data returned by the Device Server Capabilities step SECURITY PROTOCOL IN command. Management applications may ensure that their device server capabilities changes to take effect by sending a Device Server Capabilities step SECURITY PROTOCOL IN command to the device server after the changes have been made.

When the device server capabilities change (i.e., upon completion of the processing for a Device Server Capabilities step SECURITY PROTOCOL IN command that reported changed information in its parameter data), the device server shall establish a unit attention condition for the initiator port associated with every I\_T nexus except

the I\_T nexus on which the Device Server Capabilities step SECURITY PROTOCOL IN command was received (see SAM-4), with the additional sense code set to SA CREATION CAPABILITIES DATA HAS CHANGED.

The Device Server Capabilities step participates in the negotiation to skip the Authentication step as described in 5.14.4.6.

NOTE 13 - The Device Server Capabilities step has no IKEv2 exchange equivalent in RFC 4306. This step replaces most of IKEv2's negotiation by having the application client obtain the supported capabilities from the device server.

#### **5.14.4.8 IKEv2-SCSI Key Exchange step**

##### **5.14.4.8.1 Overview**

The Key Exchange step consists of a Diffie-Hellman key exchange with nonces (see RFC 4306) and is accomplished as follows:

- 1) A SECURITY PROTOCOL OUT command (see 5.14.4.8.2);
- 2) A SECURITY PROTOCOL IN command (see 5.14.4.8.3); and
- 3) Key exchange completion (see 5.14.4.8.4)

NOTE 14 - The Key Exchange step corresponds to the IKEv2 IKE\_SA\_INIT exchange in RFC 4306, except that determination of device server capabilities has been moved to the Device Server Capabilities step.

##### **5.14.4.8.2 Key Exchange step SECURITY PROTOCOL OUT command**

To send its key exchange message to the device server, the application client sends a SECURITY PROTOCOL OUT command (see 6.31) with the SECURITY PROTOCOL field set to IKEv2-SCSI (i.e., 41h) and the SECURITY PROTOCOL SPECIFIC field set to 0102h. The parameter list consists of an IKEv2-SCSI header (see 7.6.3.4) and the following:

- 1) An IKEv2-SCSI Timeout Values payload (see 7.6.3.5.14);
- 2) An IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.6.3.5.12);
- 3) If the Authentication step is skipped (see 5.14.4.1), an IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.6.3.5.13);
- 4) A Key Exchange payload (see 7.6.3.5.3); and
- 5) A Nonce payload (see 7.6.3.5.7).

The IKEv2-SCSI Timeout Values payload contains the inactivity timeouts that apply to this IKEv2-SCSI SA creation transaction (see 3.1.123) and the SA (see 3.1.148) that is created.

The IKEv2-SCSI SA Cryptographic Algorithms payload selects the cryptographic algorithms from among those returned in the Device Server Capabilities step (see 5.14.4.7) to be used in the creation of the SA.

If the Authentication step is skipped, the IKEv2-SCSI SAUT Cryptographic Algorithms payload contains the following information about the SA to be created:

- a) The cryptographic algorithms selected by the application client from among those returned in the Device Server Capabilities step; and
- b) The usage data (see 7.6.3.5.13), if any, that is specific to the SA.

If the application client is unable to select a set of algorithms that are appropriate for the intended creation and usage of the SA, the application client should not perform the Key Exchange step to request the creation of an SA.

IKEv2-SCSI SA Cryptographic Algorithms payload error checking requirements that ensure a successful negotiation of SA creation algorithms are described in 7.6.3.5.12 and 7.6.3.6.

IKEv2-SCSI SAUT Cryptographic Algorithms payload error checking requirements needed to ensure a successful SA creation are described in 7.6.3.5.13 and 7.6.3.6.

The Key Exchange payload contains the application client's Diffie-Hellman value.

The Nonce payload contains the application client's random nonce (see 3.1.115).

#### 5.14.4.8.3 Key Exchange step SECURITY PROTOCOL IN command

If the Key Exchange step SECURITY PROTOCOL OUT command (see 5.14.4.8.2) completes with GOOD status, the application client sends a SECURITY PROTOCOL IN command (see 6.30) with the SECURITY PROTOCOL field set to IKEv2-SCSI (i.e., 41h) and the SECURITY PROTOCOL SPECIFIC field set to 0102h to obtain the device server's key exchange message.

The parameter data returned by the device server in response to the SECURITY PROTOCOL IN command shall contain an IKEv2-SCSI header (see 7.6.3.4) and the following:

- 1) An IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.6.3.5.12);
- 2) If the Authentication step is skipped (see 5.14.4.1), an IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.6.3.5.13);
- 3) A Key Exchange payload (see 7.6.3.5.3);
- 4) A Nonce payload (see 7.6.3.5.7); and
- 5) Zero or more Certificate Request payloads (see 5.14.4.3).

As part of processing of the Key Exchange step SECURITY PROTOCOL IN command, the device server shall:

- a) Associate the SECURITY PROTOCOL IN command to the last Key Exchange step SECURITY PROTOCOL OUT command received on the I\_T\_L nexus. If the device server is maintaining state for at least one IKEv2-SCSI CCS and the device server is unable to establish this association, then:
  - A) The SECURITY PROTOCOL IN command shall be terminated with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS; and
  - B) The device server shall continue the IKEv2-SCSI CCS.

If the device is not maintaining state for at least one IKEv2-SCSI CSS, then the SECURITY PROTOCOL IN command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to COMMAND SEQUENCE ERROR.
- b) Return the device server's SAI in the IKEv2-SCSI header IKE\_SA DEVICE SERVER SAI field;
- c) Return the IKEv2-SCSI SA Cryptographic Algorithms payload containing:
  - A) The SA creation cryptographic algorithms supplied by the application client in the Key Exchange step SECURITY PROTOCOL OUT command parameter list; and
  - B) The device server's SAI (see 3.1.149) in the SAI field (see 7.6.3.5.12);
- d) If the Authentication step is skipped, return the IKEv2-SCSI SAUT Cryptographic Algorithms payload containing:
  - A) The SA usage cryptographic algorithms supplied by the application client in the Key Exchange step SECURITY PROTOCOL OUT command parameter list; and
  - B) The device server's SAI in the SAI field (see 7.6.3.5.13);
- e) Return information about the completed Diffie-Hellman exchange with the Key Exchange payload; and
- f) Return device server's random nonce (see 3.1.115) in the Nonce payload.

If the Key Exchange step SECURITY PROTOCOL IN command (see 5.14.4.8.3) completes with GOOD status, the application client should copy the device server's SAI from the IKE\_SA DEVICE SERVER SAI field in the IKEv2-SCSI header to the state it is maintaining for the IKEv2-SCSI CCS.

Except for the SAI field, the application client should compare the fields in the IKEv2-SCSI SA Cryptographic Algorithms payload and the IKEv2-SCSI SAUT Cryptographic Algorithms payload, if any, to the values sent in the Key Exchange step SECURITY PROTOCOL OUT command (see 5.14.4.8.2). If the application client detects differences in the contents of the payloads other than in the SAI field, the application client should abandon the IKEv2-SCSI CCS and notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 5.14.4.12.

#### 5.14.4.8.4 Key Exchange step completion

Before completing the Key Exchange step SECURITY PROTOCOL IN command (see 5.14.4.8.3) with GOOD status the device server shall complete the Key Exchange step as described in this subclause.

Upon receipt of GOOD status for the Key Exchange step SECURITY PROTOCOL IN command the application client should complete the Key Exchange step as described in this subclause.

If the Key Exchange step does not end with the IKEv2-SCSI CCS being abandoned (see 5.14.4.12), then the contents of the SA\_AUTH\_OUT IKEv2-SCSI cryptographic algorithm descriptor and SA\_AUTH\_IN IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.6) in the IKEv2-SCSI SA Cryptographic Algorithms payload specify how the shared key exchanged by the Key Exchange step SECURITY PROTOCOL OUT command and the Key Exchange step SECURITY PROTOCOL IN command is used to generate additional shared keys as follows:

- a) If the ALGORITHM IDENTIFIER field in both descriptors contain SA\_AUTH\_NONE, then the SA participants generate the SA, including the generation of the shared keys used for SA management (e.g., SA creation and management) and the shared keys used by the created SA as defined in 5.14.4.11; or
- b) If the ALGORITHM IDENTIFIER field in both descriptors contain a value other than SA\_AUTH\_NONE, then the SA participants generate shared keys (see 5.14.4.10.3) for the following:
  - A) Seeding the Authentication step generation of the shared keys used by the created SA; and
  - B) SA creation and management.

#### 5.14.4.8.5 After the Key Exchange step

Processing of the IKEv2-SCSI CCS subsequent to completion of the Key Exchange step (see 5.14.4.8.4) depends on the contents of the SA\_AUTH\_OUT IKEv2-SCSI cryptographic algorithm descriptor and SA\_AUTH\_IN IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.6) in the IKEv2-SCSI SA Cryptographic Algorithms payload as follows:

- a) If the ALGORITHM IDENTIFIER field in both descriptors contain SA\_AUTH\_NONE, then processing of the IKEv2-SCSI CCS is finished and the generated SA is ready for use; or
- b) If the ALGORITHM IDENTIFIER field in both descriptors contain a value other than SA\_AUTH\_NONE, then processing of the IKEv2-SCSI CCS continues as follows:
  - A) The Authentication step is performed (see 5.14.4.9); and
  - B) The SA participants generate the SA, including the generation of the shared keys used for SA management and the shared keys used by the created SA as defined in 5.14.4.11.

#### 5.14.4.9 IKEv2-SCSI Authentication step

##### 5.14.4.9.1 Overview

The Authentication step performs the following functions:

- a) authenticates both the application client and the device server;
- b) protects the previous steps of the protocol; and
- c) cryptographically binds the authentication and the previous steps to the created SA.

The Authentication step is accomplished as follows:

- 1) A SECURITY PROTOCOL OUT command (see 5.14.4.9.2); and
- 2) A SECURITY PROTOCOL IN command (see 5.14.4.9.3).

The parameter data for both commands shall be encrypted and integrity protected using the algorithms and keys determined in the Key Exchange step (see 5.14.4.8).

NOTE 15 - The Authentication step corresponds to the IKEv2 IKE\_AUTH exchange in RFC 4306.

##### 5.14.4.9.2 Authentication step SECURITY PROTOCOL OUT command

To send its authentication message to the device server, the application client sends a SECURITY PROTOCOL OUT command (see 6.31) with the SECURITY PROTOCOL field set to IKEv2-SCSI (i.e., 41h) and the SECURITY PROTOCOL SPECIFIC field set to 0103h. The parameter data consists of the IKEv2-SCSI header (see 7.6.3.4) and an Encrypted payload (see 7.6.3.5.10) that:

- a) Is integrity checked and encrypted as follows:
  - A) Using separate algorithms as follows:
    - a) Integrity checked using the following:
      - A) The algorithm specified by the INTEG IKEv2-SCSI algorithm descriptor (see 7.6.3.6.4) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.6.3.5.12) from the Key Exchange step (see 5.14.4.8); and
      - B) The SK\_ai shared key (see 5.14.4.4);
    - and
    - b) Encrypted using the following:
      - A) The algorithm specified by the ENCR IKEv2-SCSI algorithm descriptor (see 7.6.3.6.2) in the IKEv2-SCSI SA Cryptographic Algorithms payload from the Key Exchange step; and
      - B) The SK\_ei shared key (see 5.14.4.4);
  - or
  - B) Using a combined integrity check and encryption algorithm that uses the following (i.e., if the INTEG IKEv2-SCSI algorithm descriptor indicates AUTH\_COMBINED):
    - a) The algorithm specified by the ENCR IKEv2-SCSI algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload; and
    - b) The SK\_ei shared key with additional salt bytes as described in 5.14.4.10.1 and table 65 (see 5.14.4.4);
- and
- b) Contains the following:
  - 1) An Identification – Application Client payload (see 7.6.3.5.4);
  - 2) An IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.6.3.5.13);
  - 3) Zero or more Certificate payloads (see 5.14.4.3);
  - 4) Zero or more Certificate Request payloads (see 5.14.4.3);
  - 5) Zero or one Notify payload (see 7.6.3.5.8); and
  - 6) An Authentication payload (see 7.6.3.5.6).

Before performing any checks on data contained in the Encrypted payload, the device server shall validate the SECURITY PROTOCOL OUT command parameter data as follows:

- a) The device server shall compare the IKE\_SA APPLICATION CLIENT SAI field and the IKE\_SA DEVICE SERVER SAI field in the IKEv2-SCSI header to the SAI values it is maintaining for the IKEv2-SCSI CCS, if any, being maintained for the I\_T\_L nexus on which the SECURITY PROTOCOL OUT command was received as described in 7.6.3.4; and
- b) The device server shall decrypt and check the integrity of the Encrypted payload as described in 7.6.3.5.10.4.

Errors detected during the decryption and integrity checking of the Encrypted payload shall be handled as described in 7.6.3.8.2.

In the SECURITY PROTOCOL OUT command parameter list, the application client:

- a) Sends information about the SA usage and cryptographic algorithms;
- b) Sends its identity in the Identification – Application Client payload;
- c) Sends information proving its knowledge of the secret corresponding to its identity in the Authentication payload; and
- d) Integrity protects the Key Exchange step and the Authentication step using the Authentication payload.

The application client may include the Notify payload to send an initial contact notification to the device server. If sent, the initial contact notification specifies that the application client has no stored state for any SAs with the device server other than the SA that is being created.

In response to receipt of an initial contact notification, the device server should delete all other SAs that were authenticated with a SECURITY PROTOCOL OUT command that contained the same Identification - Application Client payload data as that which is present in the SECURITY PROTOCOL OUT command that the device server is processing.

If the device server deletes other SAs in response to an initial contact notification, it shall do so only after the successful completion of the Authentication step SECURITY PROTOCOL OUT command. If an error occurs during the Authentication SECURITY PROTOCOL OUT command, the device server shall ignore the initial contact notification.

If the device server is unable to proceed with SA creation for any reason (e.g., the verification of the Authentication payload fails), the SECURITY PROTOCOL OUT command shall be terminated with CHECK CONDITION status, with the sense key set to ABORTED COMMAND, and the additional sense code set to an appropriate value. The additional sense code AUTHENTICATION FAILED shall be used when verification of the Authentication payload fails, or when authentication fails for any other reason.

#### **5.14.4.9.3 Authentication step SECURITY PROTOCOL IN command**

If the Authentication step SECURITY PROTOCOL OUT command (see 5.14.4.9.2) completes with GOOD status, the application client sends a SECURITY PROTOCOL IN command (see 6.30) with the SECURITY PROTOCOL field set to IKEv2-SCSI (i.e., 41h) and the SECURITY PROTOCOL SPECIFIC field set to 0103h to obtain the device server's authentication message. The parameter data consists of the IKEv2-SCSI header (see 7.6.3.4) and an Encrypted payload (see 7.6.3.5.10) that:

- a) Is integrity checked using the following:
  - A) Using separate algorithms as follows:
    - a) Integrity checked using the following:



- A) The algorithm specified by the INTEG IKEv2-SCSI algorithm descriptor (see 7.6.3.6.4) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.6.3.5.12) from the Key Exchange step (see 5.14.4.8); and
  - B) The SK\_ar shared key (see 5.14.4.4);
- and
- b) Encrypted using the following:
  - A) The algorithm specified by the ENCR IKEv2-SCSI algorithm descriptor (see 7.6.3.6.2) in the IKEv2-SCSI SA Cryptographic Algorithms payload from the Key Exchange step; and
  - B) The SK\_er shared key (see 5.14.4.4);
- or
- B) Using a combined integrity check and encryption algorithm that uses the following (i.e., if the INTEG IKEv2-SCSI algorithm descriptor indicates AUTH\_COMBINED):
  - a) The algorithm specified by the ENCR IKEv2-SCSI algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload from the Key Exchange step; and
  - b) The SK\_er shared key with additional salt bytes as described in 5.14.4.10.1 and table 65 (see 5.14.4.4);
- and
- b) Contains the following:
  - 1) An Identification – Device Server payload (see 7.6.3.5.4);
  - 2) An IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.6.3.5.13);
  - 3) Zero or more Certificate payloads (see 5.14.4.3); and
  - 4) An Authentication payload (see 7.6.3.5.6).

In the SECURITY PROTOCOL IN parameter data, the device server:

- a) Confirms information about the SA usage and cryptographic algorithms;
- b) Sends its identity in the Identification – Device Server payload;
- c) Authenticates its identity; and
- d) Protects the integrity of the prior step messages using the Authentication payload.

Before completing the SECURITY PROTOCOL IN command with GOOD status, the device server shall generate the SA as described in 5.14.4.11.

The application client should verify the Authentication payload as described in 7.6.3.5.6. The Certificate payload(s) are used as part of this verification for PKI-based authentication. If the Authentication payload is verified and no other error occurs the application client should generate the SA as described in 5.14.4.11.

If the application client is unable to proceed with SA creation for any reason (e.g., the verification of the Authentication payload fails), the application client should:

- a) Not use the SA for any additional activities; and
- b) Notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 5.14.4.12.

The application client should compare the fields in the IKEv2-SCSI SAUT Cryptographic Algorithms payload to the values sent in the Authentication step SECURITY PROTOCOL OUT command (see 5.14.4.9.2). If the application client detects differences in the contents of the payloads, the application client should abandon the IKEv2-SCSI CCS and notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 5.14.4.12.

#### 5.14.4.10 Generating shared keys

##### 5.14.4.10.1 Overview

If the Authentication step is skipped (see 5.14.4.1), then shared key generation is performed as described in 5.14.4.10.2 and is summarized as follows:

- a) The shared keys for SA management (e.g., SA creation and deletion) are generated at the same time as the shared keys used by the created SA; and
- b) All the shared keys are generated during completion of the Key Exchange step (see 5.14.4.8.4).

If the Authentication step is processed (i.e., not skipped), then shared key generation is performed as described in 5.14.4.10.3 and is summarized as follows:

- a) The shared keys for SA management (e.g., SA creation and deletion) are generated during the completion of the Key Exchange step (see 5.14.4.8.4); and
- b) The shared keys used by the created SA are generated during SA generation (see 5.14.4.11).

Regardless of when the SA management shared keys (e.g., used for SA creation and deletion) and shared keys (see 5.14.4.4) used by the created SA are generated, the organization of the shared keys depends on the type of encryption and integrity checking algorithm being used as follows:

- a) If an encryption algorithm that requires separate integrity checking is used, then separate shared keys are generated for each algorithm; or
- b) If an encryption algorithm that includes integrity checking is used (i.e., if the ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.4) contains AUTH\_COMBINED), then no shared keys are generated for the integrity checking algorithm but additional key material is generated to act as a salt (see table 423 in 7.6.3.6.2) for the combined mode encryption algorithm.

##### 5.14.4.10.2 Generating shared keys when the Authentication step is skipped

If the Authentication step is skipped (see 5.14.4.1), then the shared keys for SA management are generated at the same time as the shared keys for use by the created SA.

As part of completing the Key Exchange step (see 5.14.4.8.4), the SA participants generate all necessary shared keys as follows:

- 1) Generate SKEYSEED (see RFC 4306) as described in 5.14.4.10.4;
- 2) Generate the shared keys used for SA management as described in 5.14.4.10.5;
- 3) As part of generating the SA (see 5.14.4.11) (i.e., as part of completing the Key Exchange step as described in 5.14.4.8.4), generate the shared keys for use by the created SA as described in 5.14.4.10.6 and store them in the KEYMAT SA parameter (see 3.1.126).

##### 5.14.4.10.3 Generating shared keys when the Authentication step is processed

If the Authentication step is not skipped (see 5.14.4.1), then:

- 1) The shared keys for SA management are generated during completion of the Key Exchange step (see 5.14.4.8.4) as follows:
  - 1) Generate SKEYSEED (see RFC 4306) as described in 5.14.4.10.4; and
  - 2) Generate the shared keys used for SA management as described in 5.14.4.10.5;
 and

- 2) The shared keys for use by the created SA are generated during SA generation (see 5.14.4.11), near the end of processing for the Authentication step (see 5.14.4.9) as described in 5.14.4.10.6 and store them in the KEYMAT SA parameter (see 3.1.126).

#### **5.14.4.10.4 Initializing shared key generation**

##### **5.14.4.10.4.1 Initializing for SA creation shared key generation**

The SA parameters (see 3.1.126) are initialized for the KDF function used to generate SA creation shared keys as follows:

- 1) Generate the input to the PRF function by performing the last steps of the key exchange algorithm selected by the ALGORITHM IDENTIFIER field in the D-H IKEv2-SCSI algorithm descriptor (see 7.6.3.6.5) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.6.3.5.12) using at least the contents of one of the following fields as inputs to those last steps:
  - A) The KEY EXCHANGE DATA field in the Key Exchange payload (see 7.6.3.5.3) in the Key Exchange SECURITY PROTOCOL OUT command (see 5.14.4.8.2) parameter data; and
  - B) The KEY EXCHANGE DATA field in the Key Exchange payload in the Key Exchange SECURITY PROTOCOL IN command (see 5.14.4.8.3) parameter data;
- 2) Generate SKEYSEED (see RFC 4306) using the output from step 1) and the PRF selected by the ALGORITHM IDENTIFIER field in the PRF IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.3) in the IKEv2-SCSI SA Cryptographic Algorithms payload;
- 3) Store the generated SKEYSEED value in the KEY\_SEED SA parameter;
- 4) Store the contents of the IKE\_SA APPLICATION CLIENT SAI field from the IKEv2-SCSI Header (see 7.6.3.4) from the Key Exchange step SECURITY PROTOCOL IN command (see 5.14.4.8.3) in the AC\_SAI SA parameter;
- 5) Store the contents of the IKE\_SA DEVICE SERVER SAI field from the IKEv2-SCSI Header (see 7.6.3.4) from the Key Exchange step SECURITY PROTOCOL IN command (see 5.14.4.8.3) in the DS\_SAI SA parameter;
- 6) Store the contents of the NONCE DATA field from the Nonce payload (see 7.6.3.5.7) from the parameter data for the Key Exchange step SECURITY PROTOCOL OUT command (see 5.14.4.8.2) in the AC\_NONCE SA parameter; and
- 7) Store the contents of the NONCE DATA field from the Nonce payload (see 7.6.3.5.7) from the parameter data for the Key Exchange step SECURITY PROTOCOL IN command (see 5.14.4.8.3) in the DS\_NONCE SA parameter.

##### **5.14.4.10.4.2 Initializing for generation of shared keys used by the created SA**

The SA parameters (see 3.1.126) are initialized for the KDF function used to generate the shared keys that will be used by the created SA as follows:

- 1) Store the SK\_d value that was generated along with the other shared keys used in SA creation (see 5.14.4.10.5) in the KEY\_SEED SA parameter;
- 2) Store the contents of the IKE\_SA APPLICATION CLIENT SAI field from the IKEv2-SCSI Header (see 7.6.3.4) from the Key Exchange step SECURITY PROTOCOL IN command (see 5.14.4.8.3) in the AC\_SAI SA parameter;
- 3) Store the contents of the IKE\_SA DEVICE SERVER SAI field from the IKEv2-SCSI Header (see 7.6.3.4) from the Key Exchange step SECURITY PROTOCOL IN command (see 5.14.4.8.3) in the DS\_SAI SA parameter;
- 4) Store the contents of the NONCE DATA field from the Nonce payload (see 7.6.3.5.7) from the parameter data for the Key Exchange step SECURITY PROTOCOL OUT command (see 5.14.4.8.2) in the AC\_NONCE SA parameter; and

- 5) Store the contents of the NONCE DATA field from the Nonce payload (see 7.6.3.5.7) from the parameter data for the Key Exchange step SECURITY PROTOCOL IN command (see 5.14.4.8.3) in the DS\_NONCE SA parameter.

#### 5.14.4.10.5 Generating shared keys used for SA management

The shared keys used for SA management (e.g., SA creation and deletion) are generated using the SKEYSEED generated during initialization (see 5.14.4.10.4) and the steps described in this subclause.

Which shared keys are generated for SA management depends on:

- a) Whether the encryption algorithm includes integrity checking as indicated by the contents of the ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.4) of the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.6.3.5.12), and
- b) Whether the Authentication step is skipped (see 5.14.4.1).

Using the contents of the initialized SA parameters (see 5.14.4.10.4.1), the INTEG ALGORITHM IDENTIFIER field, and the KDF selected by the ALGORITHM IDENTIFIER field in the PRF IKEv2-SCSI cryptographic algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload the following shared keys (see 5.14.4.4) are generated in the order shown:

- a) If the INTEG ALGORITHM IDENTIFIER field does not contain AUTH\_COMBINED, then generate the following shared keys (see 5.14.4.4):
  - A) If the Authentication step is not skipped, generate the following shared keys:
    - 1) SK\_d;
    - 2) SK\_ai;
    - 3) SK\_ar;
    - 4) SK\_ei;
    - 5) SK\_er;
    - 6) SK\_pi; and
    - 7) SK\_pr;

or
  - B) If the Authentication step is skipped, then generate the following shared keys:
    - 1) SK\_d;
    - 2) SK\_ai;
    - 3) SK\_ar;
    - 4) SK\_ei; and
    - 5) SK\_er.

or
- b) If the INTEG ALGORITHM IDENTIFIER field contains AUTH\_COMBINED, then generate the following shared keys:
  - A) If the Authentication step is not skipped, generate the following shared keys:
    - 1) SK\_d;
    - 2) SK\_ei with additional salt bytes as described in 5.14.4.10.1 and table 65 (see 5.14.4.4);
    - 3) SK\_er with additional salt bytes as described in 5.14.4.10.1 and table 65 (see 5.14.4.4);
    - 4) SK\_pi; and
    - 5) SK\_pr;

or
  - B) If the Authentication step is skipped, then generate the following shared keys:
    - 1) SK\_d;
    - 2) SK\_ei with additional salt bytes as described in 5.14.4.10.1 and table 65 (see 5.14.4.4); and
    - 3) SK\_er with additional salt bytes as described in 5.14.4.10.1 and table 65 (see 5.14.4.4).

The shared keys thus generated are combined with other data and stored in the MGMT\_DATA SA parameter as described in 5.14.4.11.

How to determine the sizes of the shared keys to be generated is summarized in table 65 (see 5.14.4.4).

#### 5.14.4.10.6 Generating shared keys for use by the created SA

As part of completing the Authentication step and generating the SA (see 5.14.4.11), the SA participants initialize the SA parameters for performing a KDF (see 5.14.4.10.4.2), and use the KDF selected by the ALGORITHM IDENTIFIER field in the PRF IKEv2-SCSI cryptographic algorithm descriptor in the Key Exchange step IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.6.3.5.12) to generate the following shared keys (see 5.14.4.4) in the order shown and store them in the KEYMAT SA parameter:

- a) If the ALGORITHM IDENTIFIER field in the ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.2) contains ENCR\_NULL in the IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.6.3.5.13), then generate the following shared keys:
  - 1) SK\_ai;
  - 2) SK\_ar;
 and
- b) If the ALGORITHM IDENTIFIER field in the ENCR IKEv2-SCSI cryptographic algorithm descriptor does not contain ENCR\_NULL in the IKEv2-SCSI SAUT Cryptographic Algorithms payload, then generate the following shared keys:
  - A) If the ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.4) does not contain AUTH\_COMBINED in the IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.6.3.5.13), then generate the following shared keys:
    - 1) SK\_ai;
    - 2) SK\_ar;
    - 3) SK\_ei; and
    - 4) SK\_er;
 or
  - B) If the ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor contains AUTH\_COMBINED in the IKEv2-SCSI SAUT Cryptographic Algorithms payload, then generate the following shared keys:
    - 1) SK\_ei with additional salt bytes as described in 5.14.4.10.1 and table 65 (see 5.14.4.4); and
    - 2) SK\_er with additional salt bytes as described in 5.14.4.10.1 and table 65 (see 5.14.4.4).

How to determine the sizes of the shared keys to be generated is summarized in table 65 (see 5.14.4.4).

#### 5.14.4.11 IKEv2-SCSI SA generation

Depending on whether or not the Authentication step was skipped (see 5.14.4.1), the SA participants shall generate shared keys as described in 5.14.4.10.

The SA participants shall initialize the SA parameters (see 3.1.126) as follows:

- 1) KEYMAT shall be set as follows:
  - A) If the Authentication step is skipped, KEYMAT shall be set as described in 5.14.4.10.2; or
  - B) If the Authentication step is processed, KEYMAT shall be set as described in 5.14.4.10.3;
 and
- 2) Then, the other SA parameters shall be set as follows:
  - A) AC\_SAI shall be set to the value in the IKE\_SA APPLICATION CLIENT SAI field in the IKEv2-SCSI header (see 7.6.3.4) in the Key Exchange step SECURITY PROTOCOL OUT command (see 5.14.4.8.2);
  - B) DS\_SAI shall be set to the value in the IKE\_SA DEVICE SERVER SAI field in the IKEv2-SCSI header in the Key Exchange step SECURITY PROTOCOL IN command (see 5.14.4.8.3);

- C) TIMEOUT shall be set to the IKEV2-SCSI SA INACTIVITY TIMEOUT field in the IKEv2-SCSI Timeout Values payload (see 7.6.3.5.14) in the Key Exchange step SECURITY PROTOCOL OUT command;
- D) KDF\_ID shall be set to the value in the ALGORITHM IDENTIFIER field in the PRF IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.3) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.6.3.5.12);
- E) AC\_SQN shall be set to one;
- F) DS\_SQN shall be set to one;
- G) AC\_NONCE shall be set to zero;
- H) DS\_NONCE shall be set to zero;
- I) KEY\_SEED shall be set to zero;
- J) USAGE\_TYPE shall be set to the value in the SA TYPE field in the IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.6.3.5.13) in the Key Exchange step SECURITY PROTOCOL OUT command;
- K) USAGE\_DATA shall contain at least the following values from the IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.6.3.5.13) in either the Key Exchange step SECURITY PROTOCOL OUT command or the Authentication step SECURITY PROTOCOL OUT command:
  - a) The ALGORITHM IDENTIFIER field and KEY LENGTH field from the ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.2);
  - b) The ALGORITHM IDENTIFIER field from the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.4); and
  - c) The contents, if any, of the USAGE DATA field;
 and
- L) MGMT\_DATA shall contain at least the following values:
  - a) From the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.6.3.5.12) in the Key Exchange step SECURITY PROTOCOL OUT command:
    - A) The ALGORITHM IDENTIFIER field and KEY LENGTH field from the ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.2); and
    - B) The ALGORITHM IDENTIFIER field from the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.4);
  - b) From the shared keys generated for SA management (see 5.14.4.10.5), the following shared keys and salt bytes:
    - A) The value of SK\_ai, if any;
    - B) The value of SK\_ar, if any;
    - C) The value of SK\_ei, with additional salt bytes, if any; and
    - D) The value of SK\_er, with additional salt bytes, if any;
 and
  - c) The next value of the MESSAGE ID field in the IKEv2-SCSI header.

NOTE 16 - The inclusion of the algorithm identifiers and key length in MGMT\_DATA SA parameter enables the SA to apply the same encryption and integrity algorithms that IKEv2-SCSI negotiated to future IKEv2-SCSI SECURITY PROTOCOL OUT commands and IKEv2-SCSI SECURITY PROTOCOL IN commands, if any.

#### 5.14.4.12 Abandoning an IKEv2-SCSI CCS

The occurrence of errors in either the application client or the device server may require that an IKEv2-SCSI CCS (see 3.1.66) be abandoned.

A device server shall indicate that it has abandoned an IKEv2-SCSI CCS, if any, by terminating an IKEv2-SCSI CCS command (see 5.14.4.1) received on the I\_T\_L nexus for which the IKEv2-SCSI CCS state is being maintained with any combination of status and sense data other than those shown in table 67.

**Table 67 — IKEv2-SCSI command terminations that do not abandon the CCS, if any**

<b>IKEv2-SCSI CCS Command</b>	<b>Status (Sense Key)</b>	<b>Additional Sense Code</b>	<b>Description</b>
SECURITY PROTOCOL OUT SECURITY PROTOCOL IN	GOOD (n/a)	n/a	Indicates IKEv2-SCSI CCS is progressing as described in this standard
Key Exchange step SECURITY PROTOCOL OUT (see 5.14.4.8.2)	CHECK CONDITION (ABORTED COMMAND)	CONFLICTING SA CREATION REQUEST	At least one IKEv2-SCSI CCS is already active, and attempts to start another are blocked until the first CCS completes
SECURITY PROTOCOL OUT SECURITY PROTOCOL IN	CHECK CONDITION (NOT READY)	LOGICAL UNIT NOT READY, SA CRE- ATION IN PROGRESS	Device server is busy processing another command in the IKEv2-SCSI CCS associated with this I_T_L nexus or a different IKEv2-SCSI CCS associated with a different I_T_L nexus
SECURITY PROTOCOL IN	CHECK CONDITION (ILLEGAL REQUEST)	INVALID FIELD IN CDB	Incorrect SECURITY PROTOCOL IN CDB format
Authentication step SECURITY PROTOCOL OUT (see 5.14.4.9.2)		UNABLE TO DECRYPT PARAMETER LIST	Device server is unable to decrypt the Encrypted payload (see 7.6.3.5.10) or the integrity check fails
SECURITY PROTOCOL OUT		SA CREATION PARAMETER VALUE REJECTED	To adapt to possible denial of service attacks, a condition for which the optimal response includes an additional sense code of SA CREATION PARAMETER VALUE INVALID and the abandonment of the CCS is not causing the CCS to be abandoned

As part of abandoning an IKEv2-SCSI CCS, the device server shall:

- Discard all maintained state (see 5.14.4.1); and
- Prepare to allow a future Key Exchange step SECURITY PROTOCOL OUT command received on any I\_T\_L nexus to start a new IKEv2-SCSI CCS.

After a device server abandons an IKEv2-SCSI CCS, the device server shall respond to all new IKEv2-SCSI protocol commands as if an IKEv2-SCSI CCS had never been started.

An application client should not abandon an IKEv2-SCSI CCS when the next command in the CCS is a SECURITY PROTOCOL IN command. Instead, the application client should send the appropriate SECURITY PROTOCOL IN command and then abandon the IKEv2-SCSI CCS.

An application client should specify that it has abandoned an IKEv2-SCSI CCS by sending an IKEv2-SCSI Delete operation (see 5.14.4.13) with application client SAI and device server SAI information that matches that of the IKEv2-SCSI CCS being abandoned.

#### 5.14.4.13 Deleting an IKEv2-SCSI SA

When an SA is deleted, both sets of SA parameters (see 5.14.2.2) are deleted as follows:

- 1) The application client uses the information in its SA parameters to prepare an IKEv2-SCSI Delete operation that requests deletion of the device server's SA parameters;
- 2) The application client deletes its SA parameters and any associated data;
- 3) The application client sends the IKEv2-SCSI Delete operation prepared in step 1) to the device server;
- 4) In response to the IKEv2-SCSI Delete operation, the device server deletes its SA parameters and any associated data.

The IKEv2-SCSI Delete operation is a SECURITY PROTOCOL OUT command with the SECURITY PROTOCOL field set to IKEv2-SCSI (i.e., 41h) and the SECURITY PROTOCOL SPECIFIC field set to 0104h. The parameter data consists of the IKEv2-SCSI header (see 7.6.3.4) and an Encrypted payload (see 7.6.3.5.10) that contains the one Delete payload (see 7.6.3.5.9).

The Delete payload should conform to the requirements described in 7.6.3.5.9.

The device server shall process a valid Delete operation SECURITY PROTOCOL OUT command regardless of whether or not IKEv2-SCSI CCS state is being maintained for the I\_T\_L nexus on which the command is received.

The application client SAI and device server SAI information in a valid Delete operation may or may not identify an IKEv2-SCSI CCS for which state is being maintained for the I\_T\_L nexus on which the command is received (see 7.6.3.5.10).

#### 5.14.5 Security progress indication

The cryptographic calculations required by some security protocols are capable of consuming significant amounts of time in the device server. While a cryptographic security calculations are in progress unless an error has occurred, a device server shall respond to a REQUEST SENSE command by returning parameter data containing sense data with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS with the sense key specific bytes set for progress indication (i.e., a PROGRESS INDICATION field indicating the progress of the device server in performing the necessary cryptographic calculations).

The device server shall not use the progress indication to report the detailed progress of cryptographic computations that take a variable amount of time based on their inputs. The device server may use the progress indication to report synthetic progress that does not reveal the detailed progress of the computation (e.g., divide a constant expected time for the computation by 10 and advance the progress indication in 10% increments based solely on the time).

The requirements in this subclause apply to implementations of Diffie-Hellman computations and operations involving any keys (e.g., RSA) that optimize operations on large numbers based on the values of inputs (e.g., a computational step may be skipped when a bit or set of bits in an input is zero). A progress indication that advances based on the computation structure (e.g., count of computational steps) may reveal the time taken by content-dependent portions of the computation, and reveal information about the inputs.



When cryptographic calculations are in progress, the sense data specified in this subclause shall be returned in response to a REQUEST SENSE command.

#### **5.14.6 Command security**

##### **5.14.6.1 Overview**

SCSI command security defines techniques for protecting against inadvertent or malicious misuse of SCSI commands to gain unauthorized access to logical units.

The following classes are used to specify SCSI command security:

- a) Secure CDB Originator class;
- b) Security Manager class;
- c) Enforcement Manager class; and
- d) Secure CDB Processor class.

The relationship between those classes varies depending on the implemented security technique.

##### **5.14.6.2 Secure CDB Originator class**

The Secure CDB Originator class is a kind of application client that originates SCSI commands to which it has attached a security CDB extension (see 4.3.4) that allows an enforcement manager to determine if the SCSI command may be processed by the addressed logical unit.

The secure CDB originator interacts with the security manager to determine:

- a) The types of the SCSI commands it is allowed to send to the Secure CDB processor; and
- b) The content of the security CDB extension to be attached to the SCSI commands.

##### **5.14.6.3 Secure CDB Processor class**

The Secure CDB Processor class is a kind of device server that processes SCSI commands that have an attached security extension, if an enforcement manager allows that type of SCSI command from the originating application client to be processed.

The secure CDB processor determines if a SCSI command is allowed to be processed by communicating the following information to the enforcement manager:

- a) The CDB of the SCSI command to be processed; and
- b) The security CDB extension, if any, attached to the SCSI command to be processed.

The secure CDB processor shall always allow the processing of following commands when they do not have an attached a security CDB extension:

- a) INQUIRY;
- b) REPORT LUNS;
- c) REPORT SUPPORTED OPERATION CODES;
- d) REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS;
- e) REPORT TARGET PORT GROUPS; and
- f) REQUEST SENSE.

#### 5.14.6.4 Enforcement Manager class

The Enforcement Manager class is either contained within a:

- a) Device server (i.e., has the same LUN as the secure CDB processor); or
- b) Target device (e.g., has a W\_LUN, or vendor specific presence in the SCSI target device).

The enforcement manager determines if the secure CDB processor is allowed to, or prohibited from, processing a SCSI command using security information received from the security manager.

If a secure CDB originator has not been authenticated, the enforcement manager shall not make determinations about whether a command from that secure CDB originator is permitted or prohibited.

#### 5.14.6.5 Security Manager class

The Security Manager class contains communicates with the Secure CDB Originator class (see 5.14.6.2) and the Enforcement Manager class (see 5.14.6.4) as shown in table 68.

**Table 68 — Security Manager class relationships**

Security Manager location	Communications mechanism for ...	
	Secure CDB Originator	Enforcement Manager
An application client located in the same SCSI device as the secure CDB originator	Outside the scope of this standard	Via the SCSI domain's service delivery sub-system (see SAM-4)
A device server located in the same SCSI device as the secure CDB processor	Via the SCSI domain's service delivery sub-system (see SAM-4)	Outside the scope of this standard
A SCSI device contained within the same SCSI domain as the secure CDB originator and the secure CDB processor <sup>a</sup>	Via the SCSI domain's service delivery sub-system (see SAM-4)	Via the SCSI domain's service delivery sub-system (see SAM-4)
Not a SCSI device, device server, or application client	Outside the scope of this standard	Outside the scope of this standard
<sup>a</sup> This SCSI device is required to contain an application client and a device server (i.e., to contain both a SCSI Initiator Port class (see SAM-4) and a SCSI Target Port class (see SAM-4)).		

The security manager:

- a) Maintains SCSI command security information for the SCSI domain (e.g., authorization and authentication information);
- b) Delivers to the enforcement manager the security information required by the enforcement manager to determine if the secure CDB processor is allowed to, or prohibited from, processing a SCSI command; and
- c) Responds to requests from authenticated secure CDB originators to send SCSI commands to a secure CDB processor as follows:
  - A) If the secure CDB originator sends its authentication and an authorization request, then the security manager responds with the authorization information necessary for the secure CDB originator to generate security information to be attached to any authorized CDB that is sent to the secure CDB processor; or

- B) If the secure CDB originator sends its authentication, an authorization request, and the security information to be attached to CDBs, then the security manager shall only accept the request if the secure CDB originator is authorized to send the requested SCSI commands to the requested secure CDB processor.

#### **5.14.6.6 The relationship between SAs and command security**

As defined by this standard, SAs provide the following forms of secure communications for selected portions of selected CDB and command parameter data:

- a) Cryptographic data integrity provided by Message Authentication Code or Integrity Check Value; and
- b) Confidentially provided by data encryption.

The SAs defined by this standard do not apply to data communicated in the CDBs sent from the secure CDB originator to the secure CDB processor. The function of securing CDBs is performed by the command security features described in 5.14.6. Command security and SAs features may be used in concert to protect both the CDB data and the parameter data.

Authorization information (see 5.14.6.5) includes associations between:

- a) Permissions to use certain commands and command options; and
- b) Secure CDB originator identities.

SAs provide a mechanism for satisfying the secure CDB originator authentication requirements placed on enforcement managers (see 5.14.6.4). Every secure CDB originator that is authenticated using an SA is required to be authenticated using a unique SA.

Some command security techniques (e.g., the CbCS technique described in 5.14.6.8) depend on an established secure channel between a secure CDB originator and secure CDB processor. SAs provide a mechanism for establishing such secure channels. If SAs are used in this manner, multiple secure CDB originators may share a common SA, however, such SAs do not authenticate any specific secure CDB originator.

#### **5.14.6.7 Command security techniques**

This standard defines the following techniques for implementing command security:

- a) Capability-based command security (see 5.14.6.8).

#### **5.14.6.8 Capability-based command security technique**

##### **5.14.6.8.1 Overview**

CbCS is a credential-based system that manages access to a logical unit by the coordination between shared keys and security parameters set by the CbCS management application client (see 5.14.6.8.4) and credentials generated by the CbCS management device server (see 5.14.6.8.3). The mechanism for coordination between the CbCS management device server and the CbCS management application client is not defined in this standard.

The CbCS protocol enables centralized management of SCSI command security.

CbCS secures access to a logical unit or a volume (see SSC-3) by providing cryptographic integrity of credentials that are added to commands sent to the logical unit (see 3.1.79). This cryptographic integrity is based on mutual trust and key exchanges between the CbCS management device server, CbCS management application client, and the enforcement manager (see 3.1.47).

Different levels of protection and security are achieved by using different CbCS methods. The following CbCS methods are defined by this standard:

- a) The BASIC CbCS method (see 5.14.6.8.8.2) provides protection against errors but does not prevent unauthorized access caused by means of malicious attacks (e.g., identity spoofing and network attacks); and
- b) The CAPKEY CbCS method (see 5.14.6.8.8.3) enforces application client authentication and provides cryptographic integrity of credentials. It protects against the following types of unauthorized access attacks:
  - A) Without cryptographic message integrity in the service delivery subsystem:
    - a) Illegal use of credentials beyond their original scope and life span;
    - b) Forging or stealing credentials; and
    - c) Using malformed credentials;
  - and
  - B) With cryptographic message integrity in the service delivery subsystem:
    - a) Network errors and malicious message modifications; and
    - b) Message replay attacks.

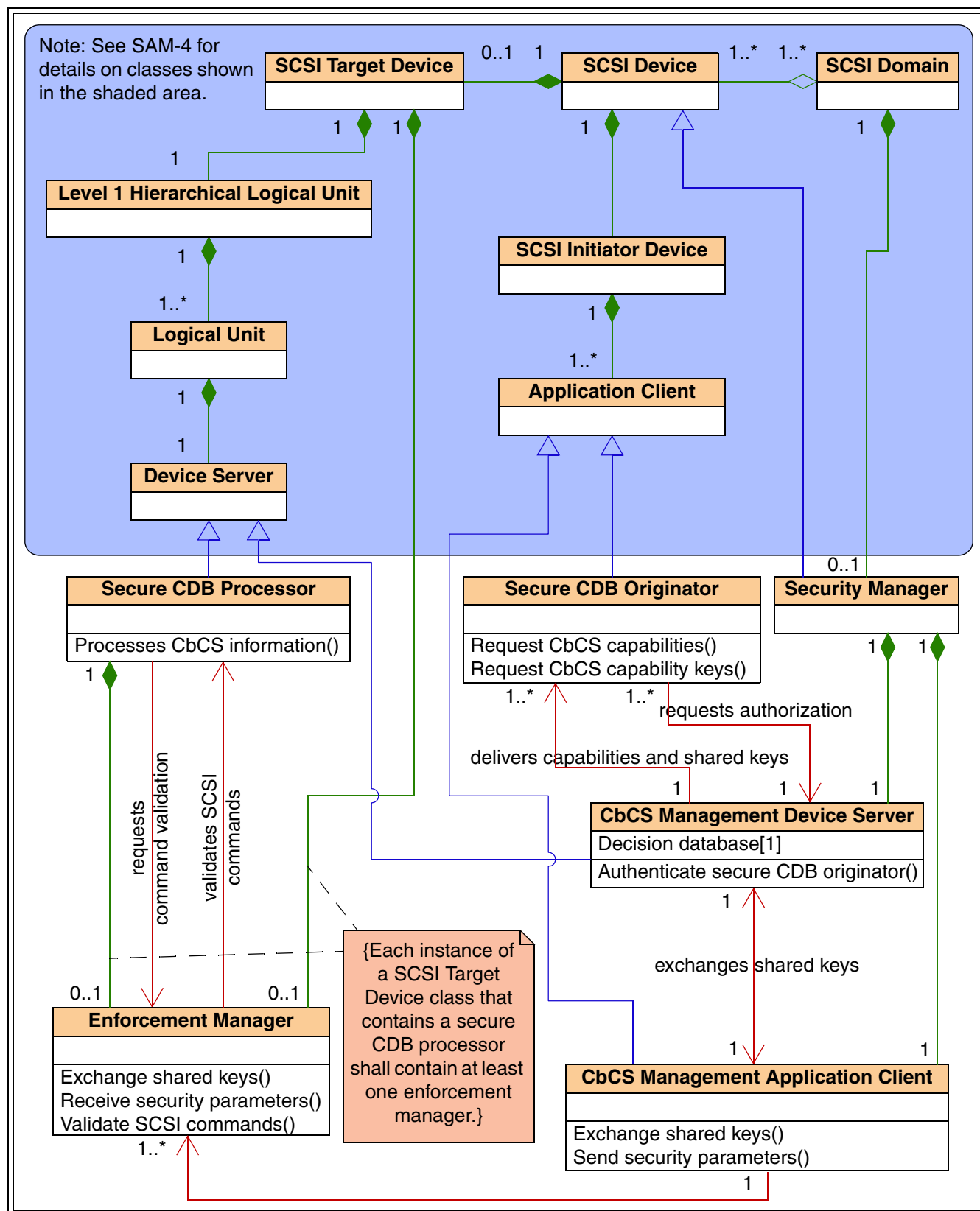
CbCS also supports rapid revocation of credentials, per SCSI target device and per logical unit.

CbCS does not define task management function security.

CbCS (see figure 15) is composed of the following class:

- a) Security Manager class (see 5.14.6.8.2) that contains the following classes:
  - A) CbCS Management Device Server class (see 5.14.6.8.3); and
  - B) CbCS Management Application Client class (see 5.14.6.8.4);
- b) SCSI Initiator Device class (see SAM-4) that contains the following class:
  - A) Secure CDB Originator class (see 5.14.6.8.5);
- and
- c) SCSI Target Device class (see SAM-4) that contains the following classes:
  - A) Secure CDB Processor class (see 5.14.6.8.6); and
  - B) Enforcement Manager class (see 5.14.6.8.7).

Figure 15 shows the flow of transactions between the components of a CbCS capable SCSI domain.



Each instance of CbCS shall contain:

- a) One security manager that shall contain:
  - A) One CbCS management device server; and
  - B) One CbCS management application;
- b) One or more SCSI initiator devices that shall contain:
  - A) One or more secure CDB originators;
 and
- c) One or more SCSI target devices that shall contain:
  - A) One secure CDB processor per logical unit; and
  - B) The following:
    - a) One enforcement manager per secure CDB processor;
    - b) One enforcement manager per SCSI target device; or
    - c) both.

#### 5.14.6.8.2 Security Manager class

The Security Manager class for the CbCS technique manages access of secure CDB originators to logical units or volumes (see SSC-3). It uses a decision database to obtain the authorization information required for deciding the type and duration of access granted to secure CDB originator to a given logical unit (see 3.1.79) or volume (see SSC-3). It communicates with secure CDB originators to provide them CbCS credentials (see 5.14.6.8.12), and with enforcement managers (see 3.1.47) to exchange shared keys (see 5.14.6.8.11) and send CbCS parameters (see 5.14.6.8.15).

The security manager may be located and may communicate with secure CDB originators and enforcement managers as follows:

- a) If it is a SCSI device contained within the same SCSI domain as the secure CDB originator and the enforcement manager, it shall contain an application client and use it to communicate to the enforcement manager, and it shall contain a device server and use it to communicate with secure CDB originators;
- b) If it is an application client located in the same device as the secure CDB originators, it shall communicate to the enforcement manager via the SCSI domain's service delivery subsystem, and it may communicate with the secure CDB originators by means outside the scope of this standard; and
- c) If it is a device server located in the same device as the secure CDB processor, it shall communicate to the secure CDB originators via the SCSI domain's service delivery subsystem, and it may communicate with the enforcement manager by means outside the scope of this standard.

The security manager's device server is called CbCS management device server (see 5.14.6.8.3). The security manager's application client is called CbCS management application client (see 5.14.6.8.4).

If the security manager is a SCSI device (see 3.1.129), it shall perform CbCS management using the CbCS management application client and the CbCS management device server as follows:

- a) The CbCS management device server (see 5.14.6.8.3) provides access policy controls to secure CDB originators (see 3.1.143) using policy-coordinated CbCS capabilities; and
- b) The CbCS management application client (see 5.14.6.8.4) prevents unsecured access to a logical unit (see 3.1.79) or a volume (see SSC-3) in concert with:
  - A) The CbCS management device server (see 5.14.6.8.3);
  - B) The enforcement manager (see 3.1.47); and
  - C) The secure CDB processor (see 3.1.144).

CbCS management is confined to the CbCS management application client and CbCS management device server. The communication of CbCS management information may occur in a manner outside the scope of this standard.

### **5.14.6.8.3 CbCS Management Device Server class**

#### **5.14.6.8.3.1 CbCS Management Device Server class overview**

The CbCS Management Device Server class returns a CbCS capability and a CbCS capability key (i.e., Capability-Key) with each CbCS credential giving the secure CDB originator access to a specific logical unit, and optionally to a volume (see SSC-3).

This standard defines the RECEIVE CREDENTIAL command (see 6.19) that the secure CDB originator may use to request a CbCS capability and a CbCS capability key from a CbCS management device server.

Commands to and responses from the CbCS management device server are protected by use of an SA whose creation included the Authentication Step (see 6.19).

The CbCS management device server shall set the CBCS bit to zero in any Extended INQUIRY Data VPD page (see 7.7.4) that it returns.

#### **5.14.6.8.3.2 Decision Database attribute**

The Decision Database attribute is used to obtain the authorization information required for deciding the type and duration of access granted to a secure CDB originator for a given logical unit (see 3.1.79) or volume (see SSC-3) within a SCSI target device. CbCS Credentials are prepared by the CbCS management device server based on the contents of that Decision Database attribute.

#### **5.14.6.8.4 CbCS Management Application Client class**

The CbCS Management Application Client class exchanges shared keys (see 5.14.6.8.11) with and sends CbCS parameters (see 5.14.6.8.15) to the enforcement manager using SECURITY PROTOCOL OUT commands (see 6.31) and SECURITY PROTOCOL IN commands (see 6.30) transferred over the SCSI domain's service delivery subsystem.

The CbCS capability keys are computed by the CbCS management device server using shared keys that are shared between the:

- a) Enforcement manager;
- b) CbCS management application client; and
- c) CbCS management device server.

The shared keys are managed by the security manager.

#### **5.14.6.8.5 Secure CDB Originator class**

The Secure CDB Originator class requests CbCS capabilities (see 3.1.16) and CbCS capability keys (see 3.1.17) from the CbCS management device server for a specific logical unit (see 3.1.79) or volume (see SSC-3). The secure CDB originator sends the CbCS capability and integrity check value to the logical unit's secure CDB processor as part of a CbCS extended CDB as described in 5.14.6.8.16.

For more information on the Secure CDB Originator class see 5.14.6.2.

#### 5.14.6.8.6 Secure CDB Processor class

The Secure CDB Processor class:

- a) Receives a CbCS capability descriptor (see 6.19.2.3) from a secure CDB originator in a CbCS extension descriptor (see 5.14.6.8.16);
- b) Requests the SCSI command be validated by the enforcement manager; and
- c) If the Enforcement Manager validates the SCSI command, then the secure CDB processor processes that SCSI command.

The secure CDB processor indicates that CbCS is applied to a logical unit by setting the CBCS bit to one in the Extended INQUIRY Data VPD page (see 7.7.4). If the CbCS bit is set to one, the logical unit shall support the following:

- a) Extended CDBs (see 4.3.4);
- b) CbCS extension type (see 5.14.6.8.16);
- c) SECURITY PROTOCOL IN commands (see 6.30) specifying the CbCS security protocol (see 7.6.4); and
- d) SECURITY PROTOCOL OUT commands (see 6.31) specifying the CbCS security protocol (see 7.6.4).

For more information on the Secure CDB Processor class see 5.14.6.3.

#### 5.14.6.8.7 Enforcement Manager class

The Enforcement Manager class:

- a) Receives shared keys (see 5.14.6.8.11) and CbCS parameters (see 5.14.6.8.15) from the CbCS management application client;
- b) Authenticates the CbCS capability received from a secure CDB processor with an integrity check value as described in 5.14.6.8.13.2;
- c) Validates SCSI commands sent by secure CDB originators as described in 5.14.6.8.13.2; and
- d) If the CAPKEY CbCS method (see 5.14.6.8.8.3) is supported, supplies one security token (see 5.14.6.8.10) for each active I\_T nexus to the secure CDB processor for delivery to the secure CDB originator that is using that I\_T nexus.

The enforcement manager may be contained within the secure CDB processor, or within the SCSI target device. If the enforcement manager is contained within the secure CDB processor then, the shared keys and CbCS parameters it uses pertain to that logical unit (see 3.1.79). If the enforcement manager is contained within the SCSI target device then, the shared keys and CbCS parameters it uses pertain to the SCSI target device, and the SECURITY PROTOCOL well-known logical unit (see 8.5) is used for the commands to exchange shared keys and set CbCS parameters.

If a shared key is stored in a well known logical unit then that key is shared between all logical units within the SCSI target device but shall only be used by a logical unit if there has been no shared key assigned to that logical unit (i.e., a shared key assigned to a logical unit always overrides any shared key assigned to a well known logical unit).

For more information on the Enforcement Manager class see 5.14.6.4.



#### 5.14.6.8.8 CbCS methods

##### 5.14.6.8.8.1 Overview

The CbCS methods defined by this standard are summarized in table 69.

**Table 69 — CbCS methods**

CbCS method	Protection provided by the enforcement manager and secure CDB processor	Level of security provided, including I_T nexus security provided by SCSI transport (e.g., FC-SP)	
		Without I_T nexus security	With I_T nexus security
BASIC	Protection against errors provided by verifying that the CbCS capability allows processing, but no validation of the authenticity of the CbCS capability.	No protection against attacks	Same protection as is provided by the SCSI transport on the I_T nexus
CAPKEY	Protection against errors and some attacks by both verifying that the CbCS capability allows processing, and validating the authenticity of the CbCS capability.	CbCS capability authenticity assured, but still subject to network attacks (e.g., replay attacks)	CbCS capability authenticity assured and bound to an I_T nexus; other network attacks (e.g., data privacy) thwarted by SCSI transport security on the I_T nexus

If a secure CDB processor receives a command for a logical unit that has CbCS enabled, the enforcement manager shall validate the command as described in 5.14.6.8.13.2 before any other field in the CDB is validated, including the operation code.

##### 5.14.6.8.8.2 The BASIC CbCS method

The BASIC CbCS method validates that the CbCS capability authorizes the encapsulated command for each CDB. It provides centrally-managed policy-driven command access control mechanism that enforces authorized access based on capabilities.

The BASIC CbCS method does not validate the authenticity of the CbCS capability.

Preparing CbCS credentials for the BASIC CbCS method does not require the knowledge of CbCS shared keys and may be done by the secure CDB originator without coordination with the CbCS management device server. In the CbCS extension descriptor (see 5.14.6.8.16):

- a) The CbCS capability descriptor (see 6.19.2.3) CBSC METHOD field is set to BASIC;
- b) The following CbCS capability descriptor fields are ignored:
  - A) The KEY VERSION field; and
  - B) The INTEGRITY CHECK VALUE ALGORITHM field;
 and
- c) The INTEGRITY CHECK VALUE field is set to zero.

The BASIC CbCS method controls access between the secure CDB originator and the secure CDB processor without requiring authentication of the secure CDB originator. It is sufficient for the secure CDB processor to request that the enforcement manager verify the CbCS capabilities sent by the secure CDB originator.

#### 5.14.6.8.3 The CAPKEY CbCS method

The CAPKEY CbCS method provides centrally-managed policy-driven command access control mechanism that enforces authorized access based on capabilities.

In addition, the CAPKEY CbCS method assures the integrity and authenticity of the CbCS capability transferred with each command.

The CAPKEY CbCS method provides for security of commands delivered to the secure CDB processor. When used in conjunction with a secure service delivery subsystem, it provides additional protection against network attacks (see 5.14.6.8.1).

Each capability (see 5.14.6.8.13) is cryptographically associated with a capability key, and the pair is returned to a secure CDB originator (see 5.14.6.8.5) in credential (see 5.14.6.8.12) in response to a RECEIVE CREDENTIAL command (see 6.19).

When the capability is associated with a specific command using a CbCS extension descriptor (see 5.14.6.8.16), an integrity check value is computed using the capability key and a CbCS security token (see 5.14.6.8.10). The enforcement manager (see 5.14.6.8.7) validates this integrity check value as described in 5.14.6.8.13.3.

#### 5.14.6.8.9 CbCS trust assumptions

After the logical unit is trusted (i.e., after a secure CDB originator authenticates that it is communicating with a specific logical unit), the secure CDB originator trusts the secure CDB processor and the enforcement manager to do the following:

- a) Deny any access attempt from any application client that is not authorized by the security manager; and
- b) Deny access from any application client that does not perform the CbCS protocols and functions defined by this standard.

The CbCS management device server and the CbCS management application client are trusted after:

- a) The CbCS management device server is authenticated by the secure CDB originator; and
- b) The CbCS management application client is authenticated by the CbCS Enforcement Manager.

The CbCS management device server and the CbCS management application client are trusted to do the following:

- a) Securely store long-lived shared keys and capability keys;
- b) Grant credentials to secure CDB originators according to access control policies that are outside the scope of this standard; and
- c) Perform the defined security functions.

The service delivery subsystem between the secure CDB originator and the secure CDB processor is not trusted. However, the CbCS security model for CbCS methods other than BASIC is defined so that commands generated by the secure CDB originator are processed by the secure CDB processor only after the secure CDB originator interacts with both the CbCS management device server and the secure CDB processor as defined in this standard.

The communications trust requirements shown in table 70 provide a basis for the CbCS trust assumptions.

**Table 70 — CbCS communications trust requirement**

For connections between	CbCS cryptographic communications trust requirements	Requirement level
secure CDB originator and secure CDB processor	Message integrity <sup>b</sup>	Optional <sup>a</sup>
secure CDB originator and CbCS management application client	Message confidentiality <sup>c</sup> and integrity <sup>b</sup>	Mandatory
CbCS management application client and enforcement manager	Message integrity <sup>b</sup>	Mandatory
CbCS management application client and CbCS management device server	Message confidentiality <sup>c</sup> and integrity <sup>b</sup>	Mandatory
<sup>a</sup> If this requirement is not met, then the conditions that table 69 (see 5.14.6.8.8) show for an I_T nexus without security apply. <sup>b</sup> Message integrity algorithms are those that table 79 (see 5.13.8) describes as integrity checking (i.e., AUTH) algorithms. <sup>c</sup> Message confidentiality algorithms are those that table 79 (see 5.13.8) describes as encryption algorithms.		

#### 5.14.6.8.10 CbCS security tokens

A CbCS security token is a random nonce (see 3.1.115) that is at least eight bytes in length that is chosen by the enforcement manager (see 5.14.6.8.7) and returned to a secure CDB originator (see 5.14.6.8.5) by the secure CDB processor (see 5.14.6.8.6) in response to a SECURITY PROTOCOL IN command (see 6.30) with the SECURITY PROTOCOL field set to 07h (i.e., CbCS) and the SECURITY PROTOCOL SPECIFIC field set to 3Fh (i.e., the Security Token CbCS page) as described in 7.6.4.3.4.

The security token shall be unique to each instance of an I\_T nexus known to the secure CDB processor and enforcement manager. Security tokens shall be reset and maintained as described in this subclause.

Each security token shall contain at least as many bytes as the largest cipher block size for all the integrity checking algorithms supported by the SCSI target device (see 7.6.4.3.3).

If a hard reset, logical unit reset, or I\_T nexus loss is detected by the secure CDB processor, the enforcement manager shall be notified of the event once for each affected I\_T nexus. In response to such a notification the enforcement manager shall discard the security token, if any, associated with the affected I\_T nexus.

After a power on, the enforcement manager shall discard all security tokens, if any, that it had been maintaining before the power on.

In response to a request for a security token for a given I\_T nexus from the secure CDB processor, the enforcement manager shall do one of the following:

- a) Return the security token value, if any, that is being maintained for the specified I\_T nexus to the secure CDB processor; or
- b) If no security token is being maintained for the specified I\_T nexus, then:
  - 1) A security token shall be prepared;
  - 2) The security token shall be returned to the secure CDB processor; and

- 3) The security token shall be maintained in association with the specified I\_T nexus until one of the events described in this subclause causes it to be discarded.

#### **5.14.6.8.11 CbCS shared keys**

##### **5.14.6.8.11.1 Overview**

Cryptographic integrity checking for CbCS capabilities depends on the following hierarchy of shared keys (see 3.1.155) that is specific to the CbCS model:

- 1) A master key that is composed of:
  - A) An authentication key; and
  - B) A generation key;and
- 2) Up to 16 working keys whose values are generated from the generation key component of the master key.

Each CbCS shared key set shall support:

- a) One master key; and
- b) At least two working keys.

Each CbCS shared key in a set has an associated identifier (see 5.14.6.8.11.2) that describes the CbCS shared key to the objects that are managing it without revealing the CbCS shared key's value.

Coordinated sets of CbCS shared keys that conform to this hierarchy are maintained by:

- a) The CbCS management application client (see 5.14.6.8.4);
- b) The CbCS management device server (see 5.14.6.8.3); and
- c) The enforcement manager (see 5.14.6.8.7).

The mechanism for coordinating CbCS shared key sets between the CbCS management application client and the CbCS management device server is outside the scope of this standard.

The following security protocols are provided by this standard for coordinating CbCS shared key sets between the CbCS management application client and the enforcement manager:

- a) A Diffie-Hellman key exchange protocol for changing all the components of the master key (see 5.14.6.8.11.4); and
- b) Mechanisms that invalidate a working key or set a working key based on the generation key component of the current master key (see 5.14.6.8.11.5).

These key management protocols associate a key identifier with each shared key (i.e., master key or working key). The CbCS management application client may use these key identifiers to describe the shared key in some way (e.g., when the shared key was last refreshed or how the intended use of the shared key). Key identifiers shall not be used to contain shared key values.

Within any SCSI target device that contains an enforcement manager, sets of CbCS shared keys are maintained as follows:

- a) A separate set of per-logical unit CbCS shared keys for a logical unit that has CbCS enabled;
- b) One target-wide set of CbCS shared keys that are accessible to all logical units that have CbCS enabled;
- or
- c) Both a target-wide set of CbCS shared keys and per-logical unit sets of CbCS shared keys.

The enforcement manager in any logical unit that has CbCS enabled shall have access to at least one set of CbCS shared keys.

If an enforcement manager has access to both a target-wide set of CbCS shared keys and a per-logical unit set of CbCS shared keys, then a working key defined in the per-logical unit set of CbCS shared keys, if any, shall be used instead of the equivalent working key from the target-wide set of CbCS shared keys.

All CbCS shared keys should be retired from active use (i.e., set, changed, or discarded) often enough to thwart key attacks. How often to retire CbCS shared keys from active use is outside the scope of this standard.

The shared keys in a CbCS shared key set and the CbCS pages used to manage them between the CbCS management application client and the enforcement manager are summarized in table 71.

**Table 71 — Summary of CbCS shared keys**

CbCS shared key	Applicable CbCS page	Data direction	Capability protection	Reference
Master Authentication, and Generation	Current CbCS Parameters <sup>a</sup>	In	Working key	7.6.4.3.5
	Set Master Key – Seed Exchange	Out	Authentication key	7.6.4.5.5
	Set Master Key – Seed Exchange	In		7.6.4.3.6
	Set Master Key – Change Master Key <sup>b</sup>	Out		7.6.4.5.6
Working key	Current CbCS Parameters <sup>a</sup>	In	Working key	7.6.4.3.5
	Invalidate Key Set Key	Out	Authentication key	7.6.4.5.3 7.6.4.5.4
<sup>a</sup> Only key identifiers are returned, not shared key values. <sup>b</sup> The new authentication key computed during the seed exchange is the authentication key used for this CbCS page.				

#### 5.14.6.8.11.2 CbCS shared key identifiers

CbCS shared key identifiers (see table 72) are set by the CbCS pages (see table 71 in 5.14.6.8.11.1) that change the values of a CbCS shared key and reported by the Current CbCS Parameters CbCS page (see 7.6.4.3.5).

**Table 72 — CbCS shared key identifier values**

Value	Description
0000 0000 0000 0000h <sup>a</sup>	The associated CbCS shared key has not been modified since the SCSI target device was manufactured
0000 0000 0000 0001h to FFFF FFFF FFFF FFFDh	The associated CbCS shared key has a valid value that has been set by the applicable CbCS page
FFFF FFFF FFFF FFFEh <sup>a</sup>	The associated CbCS shared key does not have a valid value
FFFF FFFF FFFF FFFFh <sup>a</sup>	The associated CbCS shared key is not supported
<sup>a</sup> The use of this value in the CbCS pages that change CbCS shared key values is reserved.	

#### 5.14.6.8.11.3 Specifying which CbCS shared key to change

The logical unit to which a SECURITY PROTOCOL IN command (see 6.30) or a SECURITY PROTOCOL OUT command (see 6.31) that specifies one of the CbCS pages shown in table 71 (see 5.14.6.8.11.1) is addressed determines which CbCS shared key is modified as follows:

- a) If the command is addressed to the SECURITY PROTOCOL well known logical unit (see 8.5), then the CbCS the target-wide (see 5.14.6.8.11.1) set of CbCS shared keys is modified; or
- b) If the command is addressed to a logical unit that is not a well known logical unit, then the per-logical unit (see 5.14.6.8.11.1) set of CbCS shared keys for the specified logical unit are modified.

#### 5.14.6.8.11.4 Updating a CbCS master key

Both components of the CbCS master key (i.e., the authentication key and the generation key) are changed using a single Diffie-Hellman exchange CCS (see 3.1.33) as summarized in this subclause. Use of the Diffie-Hellman exchange ensures forward secrecy of the master key.

The CbCS master key update CCS is composed of the following commands:

- 1) A SECURITY PROTOCOL OUT command processing the Set Master Key – Seed Exchange CbCS page (see 7.6.4.5.5);
- 2) A SECURITY PROTOCOL IN command processing the Set Master Key – Seed Exchange CbCS page (see 7.6.4.3.6); and
- 3) A SECURITY PROTOCOL OUT command processing the Set Master Key – Change Master Key CbCS page (see 7.6.4.5.6).

NOTE 17 - The capability key used to access the two Set Master Key – Seed Exchange CbCS pages is different from the capability key used to access the Set Master Key – Change Master Key CbCS page (see 5.14.6.8.12.3).

The device server shall maintain CCS state for only one CbCS master key update CCS for all I\_T nexuses. The command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to COMMAND SEQUENCE ERROR, if any of the following conditions occur:

- a) A command attempts to start a new CbCS master key update CCS while state is being maintained for another; or
- b) A sequence of CbCS master key update CCS commands other than the one shown in this subclause is attempted.

The device server shall discard the CbCS master key update CCS state if any of the following occur:

- a) A command in the CCS does not complete with GOOD status; or
- b) The entire CbCS master key update CCS command sequence shown in this subclause is not completed within ten seconds of the successful completion of processing for the SECURITY PROTOCOL OUT command for the Set Master Key – Seed Exchange CbCS page.

#### 5.14.6.8.11.5 Changing a CbCS working keys

A working key is invalidated using the Invalidate Key CbCS page (see 7.6.4.5.3) and set to a new value using the Set Key CbCS page (see 7.6.4.5.4).

New working keys are computed based on the applicable master key and a random number seed.

Working keys are tracked using CbCS shared key identifiers (see 5.14.6.8.11.2).

#### 5.14.6.8.12 CbCS credentials

##### 5.14.6.8.12.1 Overview

Each CbCS credential authorizes access to:

- a) A logical unit (see 3.1.79); or
- b) A specific volume (see SSC-3) mounted in a specific logical unit.

The applicability of CbCS credentials to the BASIC CbCS method (see 5.14.6.8.8.2) is outside the scope of this standard.

The format of a CbCS credential is described in 6.19.2.2.

The primary components of a CbCS credential are as follows:

- a) A CbCS capability whose format and preparation are described in 5.14.6.8.13; and
- b) The CbCS capability key that is an integrity check value (see 3.1.72) that is computed as follows:
  - A) If the credential is to be sent to a secure CDB originator (see 5.14.6.8.5), the CbCS capability key is computed based on a working key as described in 5.14.6.8.12.2; or
  - B) If the credential is being prepared for use by the CbCS management application client (see 5.14.6.8.4), the CbCS capability key is computed based on a working key or the master key as described in 5.14.6.8.12.3.

Regardless of how it is computed, the CbCS capability key is used as follows:

- a) By the secure CDB originator to prepare CbCS extension descriptors (see 5.14.6.8.16) sent to the secure CDB processor (see 5.14.6.8.6);
- b) By the CbCS management application client to prepare CbCS extension descriptors sent to the enforcement manager (see 5.14.6.8.7); and
- c) By the enforcement manager to validate the integrity of capabilities (see 5.14.6.8.13.3) in CbCS extension descriptors received by the secure CDB processor.

##### 5.14.6.8.12.2 CbCS capability key computations for the secure CDB originator

For credentials sent to the secure CDB originator (see 5.14.6.8.5), the CbCS capability key (see 5.14.6.8.12.1) is computed without knowledge of the command for which it is being prepared using the following inputs:

- a) The integrity check value algorithm specified by the INTEGRITY CHECK VALUE ALGORITHM field (see 6.19.2.3) in the CbCS capability descriptor; and
- b) The following inputs to this integrity check value algorithm:
  - A) All the bytes in the CbCS capability descriptor (see 6.19.2.3); and
  - B) The working key specified by the KEY VERSION field (see 6.19.2.3) in the CbCS capability descriptor.

##### 5.14.6.8.12.3 CbCS capability key computations for general use

The computation of the CbCS capability key depends on the command for which the CbCS capability key is being computed as described in this subclause. This computation is more general than the computation described in 5.14.6.8.12.2, but it produces the same results because the secure CDB originator should not be allowed to use the commands that produce the exceptional cases described in this subclause.

The CbCS capability key computations described in this subclause are used:

- a) For credentials that are to be used by the CbCS management application client (see 5.14.6.8.4); and
- b) By the enforcement manager (see 5.14.6.8.7) when validating a CbCS capability descriptor (see 5.14.6.8.13).

When the enforcement manager is validating a CbCS capability descriptor, the command is determined by inspecting the CDB that is being processed.

Based on the command associated with the credential, the CbCS capability key (see 5.14.6.8.12.1) is computed using the following inputs:

- a) The integrity check value algorithm specified by the INTEGRITY CHECK VALUE ALGORITHM field (see 6.19.2.3) in the CbCS capability descriptor; and
- b) The following inputs to this integrity check value algorithm:
  - A) All the bytes in the CbCS capability descriptor (see 6.19.2.3); and
  - B) The following CbCS shared key:
    - a) If the command is a SECURITY PROTOCOL IN command (see 6.30) or a SECURITY PROTOCOL OUT command (see 6.31) with the SECURITY PROTOCOL field set to 07h (i.e., CbCS) and the SECURITY PROTOCOL SPECIFIC field set to a value that is greater than CFFFh, then the following shared key is used:
      - A) If the command does not access the Set Master Key - Change Key CbCS page (see 7.6.4.5.6), the authentication key component of the master key (see 5.14.6.8.11) is used; or
      - B) If the command accesses the Set Master Key - Change Key CbCS page, then the authentication key component of the new master key (see 7.6.4.3.6) maintained in the CbCS master key update CCS (see 5.14.6.8.11.4) is used;
    - or
    - b) If the command is not one of those described in step b) B) a), then the working key specified by the KEY VERSION field (see 6.19.2.3) in the CbCS capability descriptor.

#### 5.14.6.8.13 CbCS capability descriptors

##### 5.14.6.8.13.1 Overview

CbCS capability descriptors are components of:

- a) CbCS credentials (see 5.14.6.8.12); and
- b) CbCS extension descriptors (see 5.14.6.8.16).

The format of a CbCS capability descriptor is described in 6.19.2.3.

CbCS capability descriptors contain:

- a) Information about what commands are allowed when the CbCS capability descriptor is associated with a specific CDB via a CbCS extension descriptor;
- b) Information that identifies a specific logical unit (see 3.1.79) or a specific volume (see SSC-3) mounted in a specific logical unit to which the CbCS capability is bound;
- c) An optional time limit on the validity of the CbCS capability descriptor;
- d) Information that the CbCS management application client (see 5.14.6.8.4) may use to invalidate one or more CbCS capability descriptors before the time limit expires; and
- e) Information that the enforcement manager uses to cryptographically validate the CbCS capability descriptor, if specified, as described in 5.14.6.8.13.



#### 5.14.6.8.13.2 CbCS extension descriptor validation

The enforcement manager (see 5.14.6.8.7) shall validate the CbCS capability descriptor (see 6.19.2.3) included in the CbCS extension descriptor (see 5.14.6.8.16). If the validation fails, the enforcement manager shall interact with the secure CDB processor (see 5.14.6.8.6) in a way that causes the command containing the CbCS extension descriptor to be terminated with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The enforcement manager's validation of a CbCS extension descriptor shall fail if any of the following conditions occur:

- 1) A CbCS extension descriptor is not present on a command that table 73 (see 5.14.6.8.14) shows as requiring a CbCS capability;
- 2) The command is one that table 73 (see 5.14.6.8.14) shows as never being allowed when CbCS is enabled;
- 3) The CBCS METHOD field is set to a value that is less than the value in the minimum CbCS method CbCS parameter (see 5.14.6.8.15);
- 4) The CBCS METHOD field contains a value that table 212 defines as reserved (see 6.19.2.3) or a value that the enforcement manager does not support (see 7.6.4.3.3);
- 5) If the CBCS METHOD field is set to CAPKEY and the integrity validation described in 5.14.6.8.13.3 fails;
- 6) The DESIGNATION TYPE field contains a value that table 211 defines as reserved (see 6.19.2.3);
- 7) The DESIGNATION TYPE field value is set to 1h (i.e., logical unit designation descriptor), and the contents of the DESIGNATION DESCRIPTOR field in which a logical unit name (see SAM-4) is indicated does not match the addressed logical unit;
- 8) The DESIGNATION TYPE field value is set to 2h (MAM attribute descriptor) and either of the following are true:
  - A) The ATTRIBUTE IDENTIFIER field in the DESIGNATION DESCRIPTOR field contains any value other than 0401h (i.e., MEDIUM SERIAL NUMBER); or
  - B) The DESIGNATION DESCRIPTOR field contents do not match the MAM attribute of the volume that is accessible via the addressed logical unit;
- 9) The CAPABILITY EXPIRATION TIME field contains a non-zero value and the value in the CAPABILITY EXPIRATION TIME field is less than (i.e., prior to) the current time in the clock CbCS parameter (see 5.14.6.8.15);
- 10) The POLICY ACCESS TAG field contains a non-zero value that does not match the policy access tag CbCS parameter (see 5.14.6.8.15); or
- 11) The command in the CDB field of the extended CDB (see 4.3.4) that contains the CbCS extension descriptor is not permitted by the PERMISSIONS BIT MASK field (see 5.14.6.8.14).

#### 5.14.6.8.13.3 CAPKEY CbCS method capability integrity validation

If the CbCS method is CAPKEY, the enforcement manager's validation of a CbCS capability descriptor shall fail the integrity tests if any of the described in this subclause occur.

Before attempting to cryptographically validate the integrity of the CbCS capability descriptor, the enforcement manager shall fail the validation if any of the following conditions occur in the CbCS capability descriptor (see 6.19.2.3):

- a) The KEY VERSION field specifies an invalid working key as follows:
    - A) The command is not a SECURITY PROTOCOL IN command (see 6.30) or a SECURITY PROTOCOL OUT command (see 6.31) with the SECURITY PROTOCOL field set to 07h (i.e., CbCS) and the SECURITY PROTOCOL SPECIFIC field set to a value that is greater than CFFFh (i.e., the KEY VERSION field is ignored for these commands);
    - B) The per-logical unit working key (see 5.14.6.8.11), if any, specified by the KEY VERSION field is invalid (see 5.14.6.8.11.2); and
    - C) The target-wide working key (see 5.14.6.8.11), if any, specified by the KEY VERSION field is invalid (see 5.14.6.8.11.2);
- or

- b) The INTEGRITY CHECK VALUE ALGORITHM field contains a value that is:
  - A) Not one of those that table 88 (see 5.14.8) lists as being an integrity checking (i.e., AUTH) algorithm;
  - B) Is AUTH\_COMBINED; or
  - C) Is a value that the enforcement manager does not support (see 7.6.4.3.3).

If no integrity checking configuration errors are found in the CbCS capability descriptor, the enforcement manager shall:

- 1) Compute the CbCS capability key for the CbCS capability descriptor as described in 5.14.6.8.12.3; and
- 2) Compute the expected contents of CbCS extension descriptor INTEGRITY CHECK VALUE field (see 5.14.6.8.16), using the following inputs:
  - A) The integrity check value algorithm specified by the INTEGRITY CHECK VALUE ALGORITHM field (see 6.19.2.3) in the CbCS capability descriptor; and
  - B) The following inputs to this integrity check value algorithm:
    - a) All the bytes in the security token (see 5.14.6.8.10) for the I\_T nexus on which the command was received as the string for which the integrity check value is to be computed; and
    - b) The CbCS capability key computed in step 1) as the cryptographic key.

The enforcement manager shall fail the validation if the contents of CbCS extension descriptor INTEGRITY CHECK VALUE field do not match the computed expected contents of CbCS extension descriptor INTEGRITY CHECK VALUE field.

#### 5.14.6.8.14 Association between commands and permission bits

The PERMISSIONS BIT MASK field in the CbCS capability (see 6.19.2.3) specifies which commands are allowed by the CbCS capability. When processing commands with the CbCS extension, the enforcement manager shall verify that the bits applicable to the encapsulated SCSI command are all set to one in the PERMISSIONS BIT MASK field before processing the command. The associations between commands and permission bits are specified in table 73 and table 74 for commands defined in this standard.

**Table 73 — Associations between commands and permissions (part 1 of 3)**

Command	PERMISSIONS BIT MASK bits <sup>a</sup>						
	DATA READ	DATA WRITE	PARM READ	PARM WRITE	SEC MGMT	RESRV	MGMT
ACCESS CONTROL IN	never allow <sup>b</sup>						
ACCESS CONTROL OUT	never allow <sup>b</sup>						
CHANGE ALIASES	always allow <sup>c</sup>						
EXTENDED COPY	never allow <sup>b</sup>						
INQUIRY	always allow <sup>c</sup>						
<sup>a</sup> The command in the CDB field of the extended CDB (see 4.3.4) that contains the CbCS extension descriptor shall be allowed only when all of the bits marked with a 1 in the row for that command are set in the PERMISSIONS BIT MASK field of the CbCS capability in the CbCS extension descriptor. The permissions bits represented by the empty cells in a row are ignored.							
<sup>b</sup> If the CBCS bit is set to one in the Extended INQUIRY Data VPD page (see 7.7.4), this command shall never be allowed.							
<sup>c</sup> This command shall always be allowed regardless of whether the CbCS extension descriptor is present and if the CbCS extension descriptor is present regardless of the value in the PERMISSIONS BIT MASK field.							

Table 73 — Associations between commands and permissions (part 2 of 3)

Command	PERMISSIONS BIT MASK bits <sup>a</sup>						
	DATA READ	DATA WRITE	PARM READ	PARM WRITE	SEC MGMT	RESRV	MGMT
LOG SELECT				1			
LOG SENSE			1				
MANAGEMENT PROTOCOL IN							1
MANAGEMENT PROTOCOL OUT							1
MODE SELECT(6)				1			
MODE SELECT(10)				1			
MODE SENSE(6)			1				
MODE SENSE(10)			1				
PERSISTENT RESERVE IN			1				
PERSISTENT RESERVE OUT						1	
READ ATTRIBUTE			1				
READ BUFFER					1		
READ MEDIA SERIAL NUMBER			1				
RECEIVE COPY RESULTS	never allow <sup>b</sup>						
RECEIVE CREDENTIAL	always allow <sup>c</sup>						
RECEIVE DIAGNOSTIC RESULTS			1				
REPORT ALIASES	always allow <sup>c</sup>						
REPORT IDENTIFYING INFORMATION			1				
REPORT LUNS	always allow <sup>c</sup>						
REPORT PRIORITY			1				
REPORT SUPPORTED OPERATION CODES	always allow <sup>c</sup>						
REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS	always allow <sup>c</sup>						
REPORT TARGET PORT GROUPS	always allow <sup>c</sup>						
REPORT TIMESTAMP			1				
REQUEST SENSE			1				
SECURITY PROTOCOL IN	see table 74						
SECURITY PROTOCOL OUT	see table 74						
<sup>a</sup> The command in the CDB field of the extended CDB (see 4.3.4) that contains the CbCS extension descriptor shall be allowed only when all of the bits marked with a 1 in the row for that command are set in the PERMISSIONS BIT MASK field of the CbCS capability in the CbCS extension descriptor. The permissions bits represented by the empty cells in a row are ignored.							
<sup>b</sup> If the CBCS bit is set to one in the Extended INQUIRY Data VPD page (see 7.7.4), this command shall never be allowed.							
<sup>c</sup> This command shall always be allowed regardless of whether the CbCS extension descriptor is present and if the CbCS extension descriptor is present regardless of the value in the PERMISSIONS BIT MASK field.							

**Table 73 — Associations between commands and permissions** (part 3 of 3)

Command	PERMISSIONS BIT MASK bits <sup>a</sup>						
	DATA READ	DATA WRITE	PARM READ	PARM WRITE	SEC MGMT	RESRV	MGMT
SEND DIAGNOSTIC				1			
SET IDENTIFYING INFORMATION				1			
SET PRIORITY				1			
SET TARGET PORT GROUPS				1			
SET TIMESTAMP				1	1		
TEST UNIT READY	always allow <sup>c</sup>						
WRITE ATTRIBUTE				1			
WRITE BUFFER					1		
<sup>a</sup> The command in the CDB field of the extended CDB (see 4.3.4) that contains the CbCS extension descriptor shall be allowed only when all of the bits marked with a 1 in the row for that command are set in the PERMISSIONS BIT MASK field of the CbCS capability in the CbCS extension descriptor. The permissions bits represented by the empty cells in a row are ignored. <sup>b</sup> If the CbCS bit is set to one in the Extended INQUIRY Data VPD page (see 7.7.4), this command shall never be allowed. <sup>c</sup> This command shall always be allowed regardless of whether the CbCS extension descriptor is present and if the CbCS extension descriptor is present regardless of the value in the PERMISSIONS BIT MASK field.							

The usage of the PERMISSIONS BIT MASK field for the SECURITY PROTOCOL IN command and the SECURITY PROTOCOL OUT command depend on the following characteristics as shown in table 74:

- a) The contents of the SECURITY PROTOCOL field; and
- b) The contents of the SECURITY PROTOCOL SPECIFIC field.

**Table 74 — Associations between security protocol commands and permissions**

Command	SECURITY PROTOCOL field	SECURITY PROTOCOL SPECIFIC field	Description
SECURITY PROTOCOL IN	00h	any	Always allowed regardless of whether the CbCS extension descriptor is present and if the CbCS extension descriptor is present regardless of the value in the PERMISSIONS BIT MASK field
SECURITY PROTOCOL IN	07h	0000h to 003Fh	
SECURITY PROTOCOL IN	07h	0040h to FFFFh	Allowed only if the CbCS extension descriptor is present and the SEC MGMT bit is set to one in the PERMISSIONS BIT MASK field
SECURITY PROTOCOL OUT	any	any	

Command standards (see 3.1.27) may describe the associations between commands and permission bits for the commands that they define. The processing requirements for those associations are the same as those described in this standard.

### 5.14.6.8.15 CbCS parameters

#### 5.14.6.8.15.1 Overview

CbCS parameters:

- Provide the CbCS Management Application Client class (see 5.14.6.8.4) with a means to control the operation of the Enforcement Manager class(see 5.14.6.8.7);
- Allow the Secure CDB Processor class(see 5.14.6.8.6) to receive security tokens and other CbCS information from the Enforcement Manager class; and
- Allow any application client to receive basic operational CbCS information from the Enforcement Manager class.

CbCS parameters that are not changeable indicate which CbCS features and algorithms are supported. An application client may retrieve the unchangeable CbCS parameters by using the SECURITY PROTOCOL IN command to return the Unchangeable CbCS Parameters CbCS page (see 7.6.4.3.3).

The CbCS parameters with values that change in response to various conditions are summarized in table 75.

**Table 75 — Summary of changeable CbCS parameters** (part 1 of 2)

Parameter	Support	Applicable CbCS page	Data direction	Capability protection	Reference
CbCS parameters that are updated automatically based on I_T nexus					
Security token	Optional <sup>a</sup>	Security Token	In	None	7.6.4.3.4
CbCS parameters that provide initial values for dynamically created logical units <sup>b</sup>					
Initial minimum CbCS method	Optional	Current CbCS Parameters	In	Working key	7.6.4.3.5
		Set Minimum CbCS Method <sup>c</sup>	Out	Working key	7.6.4.5.2
Initial policy access tag	Optional	Current CbCS Parameters	In	Working key	7.6.4.3.5
		Set Policy Access Tag <sup>d</sup>	Out	Working key	7.6.4.5.1
<sup>a</sup> Mandatory if the CAPKEY CbCS method (see 5.14.6.8.8.3) is supported.					
<sup>b</sup> SCSI target devices that do not dynamically create logical units may not implement these CbCS parameters. Retrieving and setting these CbCS parameters is possible only if the SECURITY PROTOCOL well known logical unit (see 8.5) is implemented. SCSI target devices that dynamically create logical units but do not implement the SECURITY PROTOCOL well known logical unit shall provide a means outside the scope of this standard for managing the values of these CbCS parameters.					
<sup>c</sup> If a Set Minimum CbCS Method CbCS page is processed by the SECURITY PROTOOCL well known logical unit, then the initial Minimum CbCS Method CbCS parameter is changed. If a Set Minimum CbCS Method CbCS page is processed by any logical unit other than the SECURITY PROTOOCL well known logical unit, then the minimum CbCS method CbCS parameter for that logical unit is changed.					
<sup>d</sup> If a Set Policy Access Tag CbCS page is processed by the SECURITY PROTOOCL well known logical unit, then the initial policy access tag CbCS parameter is changed. If a Set Policy Access Tag CbCS page is processed by any logical unit other than the SECURITY PROTOOCL well known logical unit, then the policy access tag CbCS parameter for that logical unit is changed.					

**Table 75 — Summary of changeable CbCS parameters (part 2 of 2)**

Parameter	Support	Applicable CbCS page	Data direction	Capability protection	Reference
CbCS parameters that affect the CbCS enforcement manager processing					
Minimum CbCS method	Mandatory	Current CbCS Parameters	In	Working key	7.6.4.3.5
		Set Minimum CbCS Method <sup>c</sup>	Out	Working key	7.6.4.5.2
Policy Access Tag	Mandatory	Current CbCS Parameters	In	Working key	7.6.4.3.5
		Set Policy Access Tag <sup>d</sup>	Out	Working key	7.6.4.5.1
Clock	Mandatory	Current CbCS Parameters	In	Working key	7.6.4.3.5
CbCS Shared keys and CbCS shared key identifiers	Optional <sup>a</sup>	see 5.14.6.8.11			

<sup>a</sup> Mandatory if the CAPKEY CbCS method (see 5.14.6.8.8.3) is supported.

<sup>b</sup> SCSI target devices that do not dynamically create logical units may not implement these CbCS parameters. Retrieving and setting these CbCS parameters is possible only if the SECURITY PROTOCOL well known logical unit (see 8.5) is implemented. SCSI target devices that dynamically create logical units but do not implement the SECURITY PROTOCOL well known logical unit shall provide a means outside the scope of this standard for managing the values of these CbCS parameters.

<sup>c</sup> If a Set Minimum CbCS Method CbCS page is processed by the SECURITY PROTOOCL well known logical unit, then the initial Minimum CbCS Method CbCS parameter is changed. If a Set Minimum CbCS Method CbCS page is processed by any logical unit other than the SECURITY PROTOOCL well known logical unit, then the minimum CbCS method CbCS parameter for that logical unit is changed.

<sup>d</sup> If a Set Policy Access Tag CbCS page is processed by the SECURITY PROTOOCL well known logical unit, then the initial policy access tag CbCS parameter is changed. If a Set Policy Access Tag CbCS page is processed by any logical unit other than the SECURITY PROTOOCL well known logical unit, then the policy access tag CbCS parameter for that logical unit is changed.

The security token CbCS parameter is described in 5.14.6.8.10.

The minimum CbCS method parameter indicates the minimum allowable CbCS method (see 5.14.6.8.8) to be used in capabilities (see 6.19.2.3) processed by an enforcement manager (see 5.14.6.8.7). The initial minimum CbCS method CbCS parameter provides an initial value for the minimum CbCS method CbCS parameter for dynamically created logical units.

The policy access tag parameter indicates the allowable contents of the POLICY ACCESS TAG field in capabilities (see 6.19.2.3) processed by an enforcement manager (see 5.14.6.8.7). The initial policy access tag CbCS parameter provides an initial value for the policy access tag CbCS parameter for dynamically created logical units.

The clock CbCS parameter indicates the time used by the enforcement manager (see 5.14.6.8.7) when evaluating the contents of the CAPABILITY EXPIRATION TIME field in a capability (see 6.19.2.3). The clock CbCS parameter is timestamp (see 5.13) and its value is managed using the same mechanisms that are used to manage a timestamp.

The CbCS shared keys and CbCS shared key identifiers are described in 5.14.6.8.11.

#### 5.14.6.8.16 CbCS extension descriptor format

The CbCS extension descriptor (see table 76) allows the capability-based command security technique (see 5.14.6.8) to be used with a SCSI command via the parameters specified in this subclause. Support for the CbCS extension descriptor is mandatory if CBCS bit to one in the Extended INQUIRY Data VPD page (see 7.7.4). When an extended CDB (see 4.3.4) includes a CbCS extension descriptor the CDB field may contain any CDB defined in this standard or any SCSI command standard (see 3.1.27).

**Table 76 — CbCS extension descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	EXTENSION TYPE (40h)							
1	Reserved							
3								
4	CbCS capability descriptor (see 6.19.2.3)							
75								
76	(MSB)	INTEGRITY CHECK VALUE						(LSB)
139								

The EXTENSION TYPE field contains 40h (i.e., CbCS extension descriptor).

The CbCS capability descriptor is defined in 6.19.2.3.

The contents of INTEGRITY CHECK VALUE field depend on the contents of the CBCS METHOD field in the CbCS capability descriptor as follows:

- a) If the CBCS METHOD field does not contain CAPKEY or a vendor specific value (see table 212 in 6.19.2.3), then the INTEGRITY CHECK VALUE field is reserved;
- b) If the CBCS METHOD field contains a vendor specific value, then the contents of the INTEGRITY CHECK VALUE field are vendor specific; or
- c) If the CBCS METHOD field contains CAPKEY, then the INTEGRITY CHECK VALUE field contains an integrity check value (see 3.1.72) that is computed using the integrity check value algorithm specified by the INTEGRITY CHECK VALUE ALGORITHM field in the CbCS capability descriptor (see 6.19.2.3) and the following inputs to the integrity check value algorithm:
  - A) All the bytes in the security token (see 5.14.6.8.10) for the I\_T nexus on which the command is being sent as the string for which the integrity check value is to be computed; and
  - B) The CbCS capability key from the credential (see 5.14.6.8.12) in which the CbCS capability descriptor was received by the secure CDB originator as the cryptographic key.

The enforcement manager shall validate the CbCS capability descriptor and the INTEGRITY CHECK VALUE field as described in 5.14.6.8.13.2.

### 5.14.7 ESP-SCSI for parameter data

#### 5.14.7.1 Overview

Subclause 5.14.7 defines a method for transferring encrypted and/or integrity checked parameter data in data-in buffers, data-out buffers, variable length CDBs (see 4.3.3), and extended CDBs (see 4.3.4). The method is based on the Encapsulating Security Payload (see RFC 4303) standard developed by the IETF. Because of the constrained usage of ESP-SCSI parameter data in data-in buffers and/or data-out buffers, the method defined in this standard differs from the one found in RFC 4303.

#### 5.14.7.2 ESP-SCSI required inputs

Prior to using the ESP-SCSI descriptors defined in 5.14.7, an SA shall be created (see 5.14.2.3) with SA parameters that conform to the requirements defined in 5.14.2.2 and to the following:

- a) The USAGE\_TYPE SA parameter shall be set to a value for which ESP-SCSI usage is defined in table 58 (see 5.14.2.2);
- b) The USAGE\_DATA SA parameter shall contain at least the following:
  - A) The algorithm identifier and key length for the encryption algorithm (e.g., the ALGORITHM IDENTIFIER field and KEY LENGTH field from the ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.2) in the IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.6.3.5.13) negotiated by an IKEv2-SCSI SA creation protocol (see 5.14.4)); and
  - B) The algorithm identifier for the integrity algorithm (e.g., the ALGORITHM IDENTIFIER field from the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.4) in the IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.6.3.5.13) negotiated by an IKEv2-SCSI SA creation protocol (see 5.14.4));
 and
- c) The KEYMAT SA parameter shall consist of the shared keys described in 5.14.4.10.6.

ESP-SCSI depends on the following additional information derived from the contents of the USAGE\_DATA SA parameter:

- a) The encryption algorithm identifier shall indicate:
  - A) The absence of encryption by having the value ENCR\_NULL (see table 88 in 5.14.8);
  - B) The size of the initialization vector, if any (e.g., as shown in table 423 (see 7.6.3.6.2));
  - C) The size of the salt bytes, if any (e.g., as shown in table 423 (see 7.6.3.6.2));
  - D) For combined mode encryption algorithms, the size of the integrity check value (i.e., the algorithm's MAC size as shown in table 423 (see 7.6.3.6.2));
 and
- b) The integrity algorithm identifier shall indicate:
  - A) The use of a combined mode encryption algorithm by having the value AUTH\_COMBINED (see table 88 in 5.14.8);
  - B) For non-combined mode encryption algorithms, the size of the integrity check value (e.g., as shown in table 427 (see 7.6.3.6.4)).

Each shared key in KEYMAT shall be taken from the KDF generated bits in the order shown in 5.14.4.10.6. The size of each of the shared keys in KEYMAT is determined by the negotiated encryption algorithm and integrity algorithm as described in 5.14.4.4.



### 5.14.7.3 ESP-SCSI data format before encryption and after decryption

Before data bytes are encrypted and after they are decrypted, they have the format shown in table 77.

**Table 77 — ESP-SCSI data format before encryption and after decryption**

Bit Byte	7	6	5	4	3	2	1	0
0	UNENCRYPTED BYTES							
p-1								
p	PADDING BYTES							
j-1								
j	PAD LENGTH (j-p)							
j+1	MUST BE ZERO							

The UNENCRYPTED BYTES field contains the bytes that are to be protected via encryption or that have been decrypted.

Before encryption, the PADDING BYTES field contains zero to 255 bytes. The number of padding bytes is:

- Defined by the encryption algorithm; or
- The number needed to cause the length of all bytes prior to encryption (i.e., j+2) to be a whole multiple of the alignment (see table 423 in 7.6.3.6.2) for the encryption algorithm being used.

The contents of the padding bytes are:

- Defined by the encryption algorithm; or
- If the encryption algorithm does not define the padding bytes contents, a series of one byte binary values starting at one and incrementing by one in each successive byte (i.e., 01h in the first padding byte, 02h in the second padding byte, etc.).

If the encryption algorithm does not place requirements on the contents of the padding bytes (i.e., option b) is in effect), then after decryption the contents of the padding bytes shall be verified to match the series of one byte binary values described in this subclause. If this verification is not successful in a device server, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, the additional sense code set to INVALID FIELD IN PARAMETER LIST, the SKSV bit set to one, and SENSE KEY SPECIFIC field set to indicate the last byte in the encrypted data as defined in 4.5.2.4.2. If this verification is not successful in an application client, the decrypted data should be ignored.

The PAD LENGTH field contains the number of bytes in the PADDING BYTES field.

The MUST BE ZERO field contains zero. After decryption, the contents of the MUST BE ZERO field shall be verified to be zero. If this verification is not successful in a device server, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, the additional sense code set to INVALID FIELD IN PARAMETER LIST, the SKSV bit set to one, and SENSE KEY SPECIFIC field set to indicate the last byte in the encrypted data as defined in 4.5.2.4.2. If this verification is not successful in an application client, the decrypted data should be ignored.

#### 5.14.7.4 ESP-SCSI outbound data descriptors

##### 5.14.7.4.1 Overview

When ESP-SCSI is used in a variable length CDB (see 4.3.3), an extended CDB (see 4.3.4), or parameter list data that appears in a data-out buffer, the parameter list data contains one or more descriptors selected based on the criteria shown in table 78.

**Table 78 — ESP-SCSI outbound data descriptors**

Descriptor name	External descriptor length <sup>a</sup>	Initialization vector present <sup>b</sup>	Reference
ESP-SCSI CDB	No	No	table 79 in 5.14.7.4.2
	No	Yes	table 80 in 5.14.7.4.2
ESP-SCSI data-out buffer	No	No	table 79 in 5.14.7.4.2
	No	Yes	table 80 in 5.14.7.4.2
ESP-SCSI data-out buffer without length	Yes	No	table 81 in 5.14.7.4.3
	Yes	Yes	table 82 in 5.14.7.4.3
<sup>a</sup> This is determined by the data format defined for the data-out buffer parameter data. If the format includes a length for the ESP-SCSI descriptor, then the answer to this question is yes. <sup>b</sup> This is determined from the USAGE_DATA SA parameter (see 5.14.7.2).			

#### 5.14.7.4.2 ESP-SCSI CDBs or data-out buffer parameter lists including a descriptor length

If the USAGE\_DATA SA parameter (see 5.14.7.2) indicates an encryption algorithm whose initialization vector size is zero, then the variable length CDB (see 4.3.3), extended CDB (see 4.3.4), or data-out buffer parameter list descriptor shown in table 79 contains the ESP-SCSI data.

**Table 79 — ESP-SCSI CDBs or data-out buffer parameter list descriptor without initialization vector**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)	DESCRIPTOR LENGTH (n-1)						(LSB)
1								
2		Reserved						
3								
4	(MSB)	DS_SAI						(LSB)
7								
8	(MSB)	DS_SQN						(LSB)
15								
16		ENCRYPTED OR AUTHENTICATED DATA						
i-1								
i	(MSB)	INTEGRITY CHECK VALUE						(LSB)
n								

The DESCRIPTOR LENGTH field, DS\_SAI field, DS\_SQN field, ENCRYPTED OR AUTHENTICATED DATA field, and INTEGRITY CHECK VALUE field are defined after table 80 in this subclause.

If the USAGE\_DATA SA parameter indicates an encryption algorithm whose initialization vector size (i.e.,  $s$ ) is greater than zero, the variable length CDB (see 4.3.3), extended CDB (see 4.3.4), or data-out buffer parameter data descriptor shown in table 80 contains the ESP-SCSI data.

**Table 80 — ESP-SCSI CDBs or data-out buffer full parameter list descriptor**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)	DESCRIPTOR LENGTH (n-1)						(LSB)
1								
2		Reserved						
3								
4	(MSB)	DS_SAI						(LSB)
7								
8	(MSB)	DS_SQN						(LSB)
15								
16	(MSB)	INITIALIZATION VECTOR						(LSB)
16+s-1								
16+s		ENCRYPTED OR AUTHENTICATED DATA						
i-1								
i	(MSB)	INTEGRITY CHECK VALUE						(LSB)
n								

The DESCRIPTOR LENGTH field specifies the number of bytes that follow in the ESP-SCSI CDB or ESP-SCSI data-out buffer parameter list descriptor.

The DS\_SAI field contains the value in the DS\_SAI SA parameter (see 5.14.2.2) for the SA that is being used to prepare the ESP-SCSI CDB or ESP-SCSI data-out buffer parameter list descriptor. If the DS\_SAI value is not known to the device server, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, the additional sense code set to INVALID FIELD IN PARAMETER LIST, the sksv bit set to one, and SENSE KEY SPECIFIC field set as defined in 4.5.2.4.2.

The DS\_SQN field should contain one plus the value in the application client's DS\_SQN SA parameter (see 5.14.2.2) for the SA that is being used to prepare the ESP-SCSI CDB or ESP-SCSI data-out buffer parameter list descriptor. Before sending the ESP-SCSI CDB or ESP-SCSI data-out buffer parameter list, the application client should copy the contents of the DS\_SQN field to its DS\_SQN SA parameter.

The device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, the additional sense code set to INVALID FIELD IN PARAMETER LIST, the sksv bit set to one, and SENSE KEY SPECIFIC field set as defined in 4.5.2.4.2 if any of the following conditions are detected:

- The DS\_SQN field is set to zero;
- The value in the DS\_SQN field is less than or equal to the value in the device server's DS\_SQN SA parameter; or
- The value in the DS\_SQN field is greater than 32 plus the value in the device server's DS\_SQN SA parameter.

If the DS\_SQN SA parameter is equal to FFFF FFFF FFFF FFFFh, the device server shall delete the SA.

The INITIALIZATION VECTOR field, if any, contains a value that is used as an input into the encryption algorithm and/or integrity algorithm specified by the SA specified by the DS\_SAI field. The INITIALIZATION VECTOR field is not encrypted. The encryption algorithm and/or integrity algorithm may define additional requirements for the INITIALIZATION VECTOR field.

The ENCRYPTED OR AUTHENTICATED DATA field contains:

- a) If an encryption algorithm for the SA specified by the DS\_SAI field is not ENCR\_NULL, encrypted data bytes for the following:
  - 1) The bytes in the UNENCRYPTED BYTES field (see 5.14.7.3);
  - 2) The bytes in the PADDING BYTES field (see 5.14.7.3);
  - 3) The PAD LENGTH field byte (see 5.14.7.3); and
  - 4) The MUST BE ZERO field byte (see 5.14.7.3);or
- b) Otherwise, the unencrypted data bytes.

If the integrity algorithm for the SA specified by the DS\_SAI field is AUTH\_COMBINED (see 5.14.7.2), then the AAD input to the encryption algorithm is composed of the following bytes, in order:

- 1) The bytes in the DS\_SAI field; and
- 2) The bytes in the DS\_SQN field.

The INTEGRITY CHECK VALUE field contains a value that is computed as follows:

- a) If the integrity algorithm is not AUTH\_COMBINED, the integrity check value is computed using the specified integrity algorithm with the following bytes as inputs, in order:
  - 1) The bytes in the DS\_SAI field;
  - 2) The bytes in the DS\_SQN field;
  - 3) The bytes in the INITIALIZATION VECTOR field, if any; and
  - 4) The bytes in the ENCRYPTED OR AUTHENTICATED DATA field after encryption, if any, has been performed;or
- b) If the integrity algorithm is AUTH\_COMBINED, the integrity check value is computed as an additional output of the specified encryption algorithm.

Upon receipt of ESP-SCSI CDB or ESP-SCSI data-out buffer parameter data, the device server shall compute an integrity check value for the ESP-SCSI CDB or ESP-SCSI data-out buffer parameter data as specified by the algorithms specified by the SA specified by the DS\_SAI field using the inputs shown in this subclause. If the computed integrity check value does not match the value in the INTEGRITY CHECK VALUE field, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, the additional sense code set to INVALID FIELD IN PARAMETER LIST, the SKSV bit set to one, and SENSE KEY SPECIFIC field set as defined in 4.5.2.4.2.

If the command is not terminated due to a sequence number error or a mismatch between the computed integrity check value and the contents of the INTEGRITY CHECK VALUE field, then the device server shall copy the contents of the received DS\_SQN field to its DS\_SQN SA parameter.

#### 5.14.7.4.3 ESP-SCSI data-out buffer parameter lists for externally specified descriptor length

If the USAGE\_DATA SA parameter (see 5.14.7.2) indicates an encryption algorithm whose initialization vector size is zero and the length of the ESP-SCSI data-out buffer parameter list descriptor appears elsewhere in the parameter list, then the data-out buffer parameter list descriptor shown in table 81 contains the ESP-SCSI data.

**Table 81 — ESP-SCSI data-out buffer parameter list descriptor without length and initialization vector**

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved							
3								
4	(MSB)	DS_SAI						
7								(LSB)
8	(MSB)	DS_SQN						
15								(LSB)
16	ENCRYPTED OR AUTHENTICATED DATA							
i-1								
i	(MSB)	INTEGRITY CHECK VALUE						
n								(LSB)

The DS\_SAI field, DS\_SQN field, ENCRYPTED OR AUTHENTICATED DATA field, and INTEGRITY CHECK VALUE field are defined in 5.14.7.4.2.

If the USAGE\_DATA SA parameter indicates an encryption algorithm whose initialization vector size (i.e.,  $s$ ) is greater than zero and the length of the ESP-SCSI data-out buffer parameter list descriptor appears elsewhere in the parameter list, the data-out buffer parameter list descriptor shown in table 82 contains the ESP-SCSI data.

**Table 82 — ESP-SCSI data-out buffer parameter list descriptor without length**

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved							
3								
4	(MSB)	DS_SAI						(LSB)
7								
8	(MSB)	DS_SQN						(LSB)
15								
16	(MSB)	INITIALIZATION VECTOR						(LSB)
16+s-1								
16+s		ENCRYPTED OR AUTHENTICATED DATA						(LSB)
i-1								
i	(MSB)	INTEGRITY CHECK VALUE						(LSB)
n								

The DS\_SAI field, DS\_SQN field, INITIALIZATION VECTOR field, ENCRYPTED OR AUTHENTICATED DATA field, and INTEGRITY CHECK VALUE field are defined in 5.14.7.4.2.

#### 5.14.7.5 ESP-SCSI data-in buffer parameter data descriptors

##### 5.14.7.5.1 Overview

A device server shall transfer ESP-SCSI parameter data descriptors in a data-in buffer only in response to a request that specifies an SA using the AC\_SAI SA parameter and DS\_SAI SA parameter values (see 5.14.2.2). If the specified combination of AC\_SAI and DS\_SAI values in a command that requests the transfer of ESP-SCSI parameter data descriptors is not known to the device server, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, the additional sense code set to INVALID FIELD IN PARAMETER LIST or to INVALID FIELD IN CDB, the SKSV bit set to one, and SENSE KEY SPECIFIC field set as defined in 4.5.2.4.2.

When ESP-SCSI is used in parameter data which appears in a data-in buffer, the parameter data contains one or more descriptors selected based on the criteria shown in table 83.

**Table 83 — ESP-SCSI data-in buffer parameter data descriptors**

Descriptor name	External descriptor length <sup>a</sup>	Initialization vector present <sup>b</sup>	Reference
ESP-SCSI data-in buffer	No	No	table 84 in 5.14.7.5.2
	No	Yes	table 85 in 5.14.7.5.2
ESP-SCSI data-in buffer without length	Yes	No	table 86 in 5.14.7.5.3
	Yes	Yes	table 87 in 5.14.7.5.3
<sup>a</sup> This is determined by the data format defined for the data-in buffer parameter data. If the format includes a length for the ESP-SCSI descriptor, then the answer to this question is yes. <sup>b</sup> This is determined from the USAGE_DATA SA parameter (see 5.14.7.2).			

If ESP-SCSI parameter data descriptors are used in a data-in buffer, then the outbound data (see 5.14.7.4) should include at least one ESP-SCSI descriptor using the same SA to thwart known plaintext attacks (see 5.14.1.4).

#### 5.14.7.5.2 ESP-SCSI data-in buffer parameter data including a descriptor length

If the USAGE\_DATA SA parameter (see 5.14.7.2) indicates an encryption algorithm whose initialization vector size is zero, then the data-in buffer parameter data descriptor shown in table 84 contains the ESP-SCSI data.

**Table 84 — ESP-SCSI data-in buffer parameter data descriptor without initialization vector**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)	DESCRIPTOR LENGTH (n-1)						(LSB)
1								
2		Reserved						
3								
4	(MSB)	AC_SAI						(LSB)
7								
8	(MSB)	AC_SQN						(LSB)
15								
16		ENCRYPTED OR AUTHENTICATED DATA						
i-1								
i	(MSB)	INTEGRITY CHECK VALUE						(LSB)
n								

The DESCRIPTOR LENGTH field, AC\_SAI field, AC\_SQN field, ENCRYPTED OR AUTHENTICATED DATA field, and INTEGRITY CHECK VALUE field are defined after table 85 in this subclause.



If the USAGE\_DATA SA parameter indicates an encryption algorithm whose initialization vector size (i.e., *s*) is greater than zero, the data-in buffer parameter data descriptor shown in table 85 contains the ESP-SCSI data.

**Table 85 — ESP-SCSI data-in buffer full parameter data descriptor**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)	DESCRIPTOR LENGTH (n-1)						(LSB)
1								
2		Reserved						
3								
4	(MSB)	AC_SAI						(LSB)
7								
8	(MSB)	AC_SQN						(LSB)
15								
16	(MSB)	INITIALIZATION VECTOR						(LSB)
16+s-1								
16+s		ENCRYPTED OR AUTHENTICATED DATA						
i-1								
i	(MSB)	INTEGRITY CHECK VALUE						(LSB)
n								

The DESCRIPTOR LENGTH field specifies the number of bytes that follow in the ESP-SCSI data-in buffer parameter data descriptor.

The AC\_SAI field contains the value in the AC\_SAI SA parameter (see 5.14.2.2) for the SA that is being used to prepare the ESP-SCSI data-in buffer parameter data descriptor. If the AC\_SAI value is not known to the application client, the ESP-SCSI data-in parameter data descriptor should be ignored.

The AC\_SQN field contains one plus the value in the device server's AC\_SQN SA parameter (see 5.14.2.2) for the SA that is being used to prepare the ESP-SCSI data-on buffer parameter data descriptor. Before sending the ESP-SCSI data-out buffer parameter list as part of a command that completes with GOOD status, the device server shall copy the contents of the AC\_SQN field to its AC\_SQN SA parameter. The device server shall not send two ESP-SCSI data-out buffer parameter data descriptors that contain the same values in AC\_SAI field and AC\_SQN field.

If the AC\_SQN SA parameter is equal to FFFF FFFF FFFF FFFFh, the device server shall delete the SA after the data-in buffer parameter data containing that value is sent.

The application client should ignore the ESP-SCSI data-in parameter data descriptor if any of the following conditions are detected:

- The AC\_SQN field is set to zero;
- The value in the AC\_SQN field is less than or equal to the value in the application client's AC\_SQN SA parameter; or
- The value in the AC\_SQN field is greater than 32 plus the value in the application client's AC\_SQN SA parameter.

The INITIALIZATION VECTOR field, if any, contains a value that is used as an input into the encryption algorithm and/or integrity algorithm specified by the SA specified by the AC\_SAI field. The INITIALIZATION VECTOR field is not encrypted. The encryption algorithm and/or integrity algorithm may define additional requirements for the INITIALIZATION VECTOR field.

The ENCRYPTED OR AUTHENTICATED DATA field contains:

- a) If an encryption algorithm specified by the SA specified by the AC\_SAI field is not ENCR\_NULL, encrypted data bytes for the following:
  - 1) The bytes in the UNENCRYPTED BYTES field (see 5.14.7.3);
  - 2) The bytes in the PADDING BYTES field (see 5.14.7.3);
  - 3) The PAD LENGTH field byte (see 5.14.7.3); and
  - 4) The MUST BE ZERO field byte (see 5.14.7.3);or
- b) Otherwise, the unencrypted data bytes.

If the integrity algorithm for the SA specified by the AC\_SAI field is AUTH\_COMBINED (see 5.14.7.2), then the AAD input to the encryption algorithm is composed of the following bytes, in order:

- 1) The bytes in the AC\_SAI field; and
- 2) The bytes in the AC\_SQN field;

The INTEGRITY CHECK VALUE field contains a value that is computed as follows:

- a) If the integrity algorithm is not AUTH\_COMBINED, the integrity check value is computed using the specified integrity algorithm with the following bytes as inputs, in order:
  - 1) The bytes in the AC\_SAI field;
  - 2) The bytes in the AC\_SQN field;
  - 3) The bytes in the INITIALIZATION VECTOR field, if any; and
  - 4) The bytes in the ENCRYPTED OR AUTHENTICATED DATA field after encryption, if any, has been performed;or
- b) If the integrity algorithms is AUTH\_COMBINED, the integrity check value is computed as an additional output of the specified encryption algorithm.

Upon receipt of ESP-SCSI data-in buffer parameter data, the application client should compute an integrity check value for the ESP-SCSI parameter data as specified by the algorithms specified by the SA specified by the AC\_SAI field using the inputs shown in this subclause. If the computed integrity check value does not match the value in the INTEGRITY CHECK VALUE field, the results returned by the command should be ignored.

The application client should copy the contents of the AC\_SQN field to its AC\_SQN SA parameter if all of the following are true:

- a) The command completed with GOOD status;
- b) The ESP-SCSI data-in parameter data descriptor was not ignored due to inconsistency problems with the AC\_SQN field; and
- c) The computed integrity check value matched the contents of the INTEGRITY CHECK VALUE field.

### 5.14.7.5.3 ESP-SCSI data-in buffer parameter data for externally specified descriptor length

If the USAGE\_DATA SA parameter (see 5.14.7.2) indicates an encryption algorithm whose initialization vector size is zero and the length of the ESP-SCSI data-in buffer parameter data descriptor appears elsewhere in the parameter data, then the data-in buffer parameter data descriptor shown in table 86 contains the ESP-SCSI data.

**Table 86 — ESP-SCSI data-in buffer parameter data descriptor without length and initialization vector**

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved							
3								
4	(MSB)	AC_SAI						
7								(LSB)
8	(MSB)	AC_SQN						
15								(LSB)
16	ENCRYPTED OR AUTHENTICATED DATA							
i-1								
i	(MSB)	INTEGRITY CHECK VALUE						
n								(LSB)

The AC\_SAI field, AC\_SQN field, ENCRYPTED OR AUTHENTICATED DATA field, and INTEGRITY CHECK VALUE field are defined in 5.14.7.5.2.

If the USAGE\_DATA SA parameter indicates an encryption algorithm whose initialization vector size (i.e.,  $s$ ) is greater than zero and the length of the ESP-SCSI data-in buffer parameter data descriptor appears elsewhere in the parameter data, the data-in buffer parameter data descriptor shown in table 87 contains the ESP-SCSI data.

**Table 87 — ESP-SCSI data-in buffer parameter data descriptor without length**

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved							
3								
4	(MSB)	AC_SAI						(LSB)
7								
8	(MSB)	AC_SQN						(LSB)
15								
16	(MSB)	INITIALIZATION VECTOR						(LSB)
16+s-1								
16+s		ENCRYPTED OR AUTHENTICATED DATA						(LSB)
i-1								
i	(MSB)	INTEGRITY CHECK VALUE						(LSB)
n								

The AC\_SAI field, AC\_SQN field, INITIALIZATION VECTOR field, ENCRYPTED OR AUTHENTICATED DATA field, and INTEGRITY CHECK VALUE field are defined in 5.14.7.5.2.

### 5.14.8 Security algorithm codes

Table 88 lists the security algorithm codes used in security protocol parameter data.

**Table 88 — Security algorithm codes** (part 1 of 2)

Code	Description	Reference
Encryption algorithms		
0001 000Ch	CBC-AES-256-HMAC-SHA-1	IEEE 1619.1
0001 0010h	CCM-128-AES-256	IEEE 1619.1
0001 0014h	GCM-128-AES-256	IEEE 1619.1
0001 0016h	XTS-AES-256-HMAC-SHA-512	IEEE 1619.1
8001 000Bh <sup>a</sup>	ENCR_NULL	7.6.3.6.2
8001 000Ch <sup>a</sup>	AES-CBC	RFC 3602
8001 0010h <sup>a</sup>	AES-CCM with a 16 byte MAC	RFC 4309
8001 0014h <sup>a</sup>	AES-GCM with a 16 byte MAC	RFC 4106
8001 0400h to 8001 FFFFh	Vendor specific	
PRF and KDF algorithms <sup>b</sup>		
8002 0002h <sup>a</sup>	IKEv2-use based on SHA-1	table 425 (see 7.6.3.6.3)
8002 0004h <sup>a</sup>	IKEv2-use based on AES-128 in CBC mode	
8002 0005h <sup>a</sup>	IKEv2-use based on SHA-256	
8002 0007h <sup>a</sup>	IKEv2-use based on SHA-512	
8002 0400h to 8002 FFFFh	Vendor specific	
Integrity checking (i.e., AUTH) algorithms		
8003 0002h <sup>a</sup>	AUTH_HMAC_SHA1_96	RFC 2404
8003 000Ch <sup>a</sup>	AUTH_HMAC_SHA2_256_128	RFC 4868
8003 000Eh <sup>a</sup>	AUTH_HMAC_SHA2_512_256	RFC 4868
F003 0000h	AUTH_COMBINED	7.6.3.6.4
8003 0400h to 8003 FFFFh	Vendor specific	
<sup>a</sup> The lower order 16 bits of this code value are assigned to match an IANA assigned value, if any, for an equivalent IKEv2 encryption algorithm (see 3.1.73) and values of 800xh in the high order 16 bits have x selected to match the IANA assigned IKEv2 transform type (e.g., 8001h – Encryption Algorithms, 8002h – PRFs and KDFs). <sup>b</sup> PRFs are equivalent to the prf() functions defined in RFC 4306. KDFs are equivalent to the prf+() functions defined in RFC 4306. <sup>c</sup> The low order 8 bits of this code value are assigned to match the auth method field in the Authentication payload (see 7.6.3.5.6).		

**Table 88 — Security algorithm codes (part 2 of 2)**

Code	Description	Reference
Diffie-Hellman algorithms		
8004 000Eh <sup>a</sup>	2 048-bit MODP group (finite field D-H)	RFC 3526
8004 000Fh <sup>a</sup>	3 072-bit MODP group (finite field D-H)	RFC 3526
8004 0010h <sup>a</sup>	4 096-bit MODP group (finite field D-H)	RFC 3526
8004 0013h <sup>a</sup>	256-bit random ECP group	RFC 4753
8004 0015h <sup>a</sup>	521-bit random ECP group	RFC 4753
8004 0400h to 8004 FFFFh	Vendor specific	
SA Authentication payload authentication algorithms		
00F9 0000h <sup>c</sup>	SA_AUTH_NONE	5.14.4.6 and 7.6.3.6.6
00F9 0001h <sup>c</sup>	RSA Digital Signature with SHA-1	RFC 4306
00F9 0002h <sup>c</sup>	Shared Key Message Integrity Code	RFC 4306
00F9 0009h <sup>c</sup>	ECDSA with SHA-256 on the P-256 curve	RFC 4754
00F9 000Bh <sup>c</sup>	ECDSA with SHA-512 on the P-521 curve	RFC 4754
00F9 00C9h to 00F9 00FFh	Vendor specific	
Other algorithms		
0000 0000h to 0000 FFFFh	Restricted	IANA
All other values	Reserved	
<sup>a</sup> The lower order 16 bits of this code value are assigned to match an IANA assigned value, if any, for an equivalent IKEv2 encryption algorithm (see 3.1.73) and values of 800xh in the high order 16 bits have x selected to match the IANA assigned IKEv2 transform type (e.g., 8001h – Encryption Algorithms, 8002h – PRFs and KDFs). <sup>b</sup> PRFs are equivalent to the prf() functions defined in RFC 4306. KDFs are equivalent to the prf+() functions defined in RFC 4306. <sup>c</sup> The low order 8 bits of this code value are assigned to match the auth method field in the Authentication payload (see 7.6.3.5.6).		

## 5.15 Identifying information

The REPORT IDENTIFYING INFORMATION command (see 6.22) and SET IDENTIFYING INFORMATION command (see 6.33) allow an application client to maintain one or more sets of identifying information associated with the peripheral device.

Identifying information shall persist through power cycles (i.e., be stored in non-volatile storage), hard resets, logical unit resets, I\_T nexus losses, media format operations, and media replacement.

Table 89 defines the identifying information types.

**Table 89 — Identifying information types**

Code	Description	Length	Support <sup>a</sup>
0000000b	Peripheral device identifying information – a value describing the peripheral device (e.g., an operating system volume label).	0 to 64 bytes	Mandatory
		65 to 512 bytes	Optional
0000010b	Peripheral device text identifying information – a null-terminated (see 4.4.2) UTF-8 (see 3.1.180) format string providing an informational description of the peripheral device (e.g., a descriptive string entered by a system administrator).	0 to 256 bytes	Optional
xxxxxx1b	Restricted (see SCC-2)		
All other values	Reserved		
<sup>a</sup> These support requirements shall apply only if the REPORT IDENTIFYING INFORMATION command and/or SET IDENTIFYING INFORMATION command are implemented.			

Identifying information is changed by:

- a) The SET IDENTIFYING INFORMATION command; or
- b) A mechanism outside the scope of this standard (e.g., a system administrator may be able to change identifying information through a management interface).

## 5.16 Downloading and activating microcode

SCSI target device implementations may use microcode (e.g., firmware) that is stored in nonvolatile storage. Microcode may be changeable by an application client using the WRITE BUFFER command (see 6.39). The WRITE BUFFER command provides multiple methods for downloading microcode to the SCSI target device and activating the microcode.

Downloading and activating microcode involves the following steps:

- 1) **Download:** The application client transfers microcode from the Data-Out buffer to the device server in one or more WRITE BUFFER commands;
- 2) **Save:** After receiving the complete microcode, if defined by the download microcode mode, the device server saves the microcode to nonvolatile storage; and
- 3) **Activate:** After receiving the complete microcode and after saving it to nonvolatile storage if defined by the download microcode mode, the SCSI target device begins using the new microcode for the first time after an event defined by the download microcode mode.

After power on or hard reset, the SCSI target device shall use the last microcode that was saved to nonvolatile storage.

Table 90 defines the WRITE BUFFER download microcode modes with respect to the steps described in this subclause.

**Table 90 — WRITE BUFFER download microcode modes**

Mode	Down- load <sup>a</sup>	Save <sup>b</sup>	Activate <sup>c</sup>
Download microcode and activate (i.e., 04h)	yes <sup>d</sup>	no	yes
Download microcode, save, and activate (i.e., 05h)	yes <sup>d</sup>	yes	optional
Download microcode with offsets and activate (i.e., 06h)	yes <sup>e</sup>	no	yes
Download microcode with offsets, save, and activate (i.e., 07h)	yes <sup>e</sup>	yes	optional
Download microcode with offsets, save, and defer activate (i.e., 0Eh) <sup>f</sup>	yes <sup>e</sup>	yes	no
Activate deferred microcode (i.e., 0Fh)	no	no	yes

<sup>a</sup> Entries in the Download column are as follows. For modes labeled yes, the application client delivers microcode in the WRITE BUFFER command(s). For modes labeled no, the application client does not deliver microcode with the WRITE BUFFER command.

<sup>b</sup> Entries in the Save column are as follows. For modes labeled yes, the device server shall save the microcode to nonvolatile storage for use after each subsequent power on or hard reset, and shall not return GOOD status for the final command in the WRITE BUFFER sequence (i.e., the series of WRITE BUFFER commands that downloads the microcode) until the microcode has been saved. For modes labeled no, the device server shall discard the microcode on the next power on or hard reset.

<sup>c</sup> Entries in the Activate column are as follows. For modes labeled yes, the device server shall activate the microcode after completion of the final command in the WRITE BUFFER sequence (i.e., the series of WRITE BUFFER commands that downloads the microcode and activates it). For modes labeled optional, the device server may or may not activate the microcode image upon completion of the final command in the WRITE BUFFER sequence. For modes labeled no, the device server shall not activate the microcode upon completion of the final command in the WRITE BUFFER sequence.

<sup>d</sup> The application client delivers microcode in a vendor specific number of WRITE BUFFER commands (i.e., a WRITE BUFFER sequence). The device server should require that the microcode be delivered in a single command. The device server shall perform any required verification of the microcode prior to returning GOOD status for the final WRITE BUFFER command in a sequence (i.e., the WRITE BUFFER command delivering the last part of the microcode).

<sup>e</sup> The application client delivers microcode in one or more WRITE BUFFER commands, specifying a buffer ID and buffer offset in each command. If the device server does not receive the necessary WRITE BUFFER commands required to deliver the complete microcode before a logical unit reset occurs, an I\_T nexus loss occurs, or a WRITE BUFFER command specifying a different download microcode mode is processed, the device server shall discard the new microcode. If the device server determines that it is processing the final WRITE BUFFER command (i.e., the WRITE BUFFER command delivering the last part of the microcode), it shall perform any required verification of the microcode prior to returning GOOD status for the command.

<sup>f</sup> Microcode downloaded with this mode is defined as deferred microcode.

When microcode is activated due to processing a WRITE BUFFER command with a mode that causes activation after processing (see table 90), the device server shall establish a unit attention condition (see SAM-4) for the initiator port associated with every I\_T nexus except the I\_T nexus on which the WRITE BUFFER command was received with the additional sense code set to MICROCODE HAS BEEN CHANGED. The application client on the I\_T nexus on which the WRITE BUFFER command was transferred should respond to GOOD status for the WRITE



BUFFER command the same way that it responds to a unit attention condition with an additional sense code set to MICROCODE HAS BEEN CHANGED (e.g., assume a hard reset has occurred).

When microcode is activated due to processing a WRITE BUFFER command with a mode that optionally causes activation after processing (see table 90), the device server shall establish a unit attention condition (see SAM-4) for the initiator port associated with every I\_T nexus with the additional sense code set to MICROCODE HAS BEEN CHANGED.

When microcode is activated due to power on or hard reset, the device server may establish a unit attention condition (see SAM-4) for the initiator port associated with every I\_T nexus with the additional sense code set to MICROCODE HAS BEEN CHANGED in addition to the unit attention condition for the power on or hard reset.

When deferred microcode (see table 90) is activated due to a command defined by its command standard as causing deferred microcode to be activated (e.g., the FORMAT UNIT command and the START STOP UNIT command in SBC-3), the device server:

- a) Shall establish a unit attention condition (see SAM-4) for the initiator port associated with every I\_T nexus with the additional sense code set to MICROCODE HAS BEEN CHANGED; and
- b) May establish other unit attention condition(s) as defined for the command (e.g., CAPACITY DATA HAS CHANGED for the FORMAT UNIT command).

If new microcode is saved before deferred microcode is activated, the deferred microcode is not activated and the new saved microcode is considered deferred.

Table 91 summarizes how the WRITE BUFFER download microcode modes process the BUFFER ID field, the BUFFER OFFSET field, and the PARAMETER LIST LENGTH field in the WRITE BUFFER CDB.

**Table 91 — WRITE BUFFER download microcode field processing**

<b>Mode</b>	<b>BUFFER ID field, BUFFER OFFSET field, and PARAMETER LIST LENGTH field processing</b>
Download microcode and activate (i.e., 04h)	vendor specific
Download microcode, save, and activate (i.e., 05h)	vendor specific
Download microcode with offsets and activate (i.e., 06h)	6.39.7
Download microcode with offsets, save, and activate (i.e., 07h)	6.39.7
Download microcode with offsets, save, and defer activate (i.e., 0Eh)	6.39.7
Activate deferred microcode (i.e., 0Fh)	ignored

If the device server is unable to process a WRITE BUFFER command with a download microcode mode because of a vendor specific condition (e.g., the device server requires the microcode be delivered in order, and the BUFFER OFFSET field is not equal to the contents of the previous WRITE BUFFER command's BUFFER OFFSET field plus the contents of the previous WRITE BUFFER command's PARAMETER LIST LENGTH field), it shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to COMMAND SEQUENCE ERROR.

The MULTI I\_T NEXUS MICROCODE DOWNLOAD field in the Extended INQUIRY Data VPD page (see 7.7.4) indicates how the device server handles concurrent attempts to download microcode using the WRITE BUFFER command download microcode modes (see table 90) from multiple I\_T nexuses.

**Table 92 — Multiple I\_T nexus handling for WRITE BUFFER download microcode modes (part 1 of 2)**

MULTI I_T NEXUS MICROCODE DOWNLOAD field <sup>a</sup>	Description
0h	The handling of concurrent WRITE BUFFER download microcode operations from multiple I_T nexus is vendor specific.
1h	<p>For modes that download microcode (see table 90), the device server shall:</p> <ul style="list-style-type: none"> <li>a) If a WRITE BUFFER command with the BUFFER OFFSET field set to zero is received on any I_T nexus, the command shall be processed as described elsewhere in this subclause. This shall establish the I_T nexus for the WRITE BUFFER sequence, and cause any microcode downloaded on another I_T nexus to be discarded;</li> <li>b) If a WRITE BUFFER command with the BUFFER OFFSET field set to a non-zero value is received on the established I_T nexus for the WRITE BUFFER sequence, the command shall be processed as described elsewhere in this subclause; and</li> <li>c) If a WRITE BUFFER command with the BUFFER OFFSET field set to a non-zero value is received on an I_T nexus that is different from the established I_T nexus for the WRITE BUFFER sequence, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to COMMAND SEQUENCE ERROR.</li> </ul> <p>For mode 0Fh (i.e., activate deferred microcode), the device server shall terminate any WRITE BUFFER command with mode 0Eh (i.e., download with offsets, save, and defer activate mode) received on an I_T nexus that is different from the established I_T nexus for the WRITE BUFFER sequence with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to COMMAND SEQUENCE ERROR.</p>
2h	<p>For modes that download microcode (see table 90), the device server shall allow concurrent sequences of WRITE BUFFER commands to be processed as described elsewhere in this subclause on more than one I_T nexus.</p> <p>For mode 0Fh (i.e., activate deferred microcode), the device server shall allow a WRITE BUFFER command with mode 0Eh (i.e., download with offsets, save, and defer activate mode) to be processed as described elsewhere in this subclause on an I_T nexus that is different from the established I_T nexus of the WRITE BUFFER sequence.</p>
<sup>a</sup> This field is contained in the Extended INQUIRY Data VPD page (see 7.7.4).	

**Table 92 — Multiple I\_T nexus handling for WRITE BUFFER download microcode modes (part 2 of 2)**

MULTI I_T NEXUS MICROCODE DOWNLOAD field <sup>a</sup>	Description
3h	<p>For modes that download microcode (see table 90), the device server shall:</p> <ul style="list-style-type: none"> <li>a) If a WRITE BUFFER command with the BUFFER OFFSET field set to zero is received on any I_T nexus, the command shall be processed as described elsewhere in this subclause. This shall establish the I_T nexus for the WRITE BUFFER sequence, and cause any microcode downloaded on another I_T nexus to be discarded;</li> <li>b) If a WRITE BUFFER command with the BUFFER OFFSET field set to a non-zero value is received on the established I_T nexus for the WRITE BUFFER sequence, the command shall be processed as described elsewhere in this subclause; and</li> <li>c) If a WRITE BUFFER command with the BUFFER OFFSET field set to a non-zero value is received on an I_T nexus that is different from the established I_T nexus for the WRITE BUFFER sequence, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to COMMAND SEQUENCE ERROR.</li> </ul> <p>For mode 0Fh (i.e., activate deferred microcode), the device server shall allow a WRITE BUFFER command with mode 0Eh (i.e., download with offsets, save, and defer activate mode) to be processed as described elsewhere in this subclause on an I_T nexus that is different from the established I_T nexus for the WRITE BUFFER sequence.</p>
4h to Fh	Reserved
<sup>a</sup> This field is contained in the Extended INQUIRY Data VPD page (see 7.7.4).	

For all WRITE BUFFER command modes that download microcode (see table 90), the COMMAND SPECIFIC field (see 6.25.4.2) located in the command timeouts descriptor of the parameter data returned by the REPORT SUPPORTED OPERATION CODES command (see 6.25) indicates the maximum time that access to the SCSI device is limited or not possible through any SCSI ports associated with a logical unit that processes a WRITE BUFFER command that activates microcode.

## 6 Commands for all device types

### 6.1 Summary of commands for all device types

The operation codes for commands that apply to all device types when the MCHNGR bit is set to zero, the SCCS bit is set to zero, and the ENCSERV bit is set to zero in the standard INQUIRY data (see 6.4.2) are listed in table 93.

**Table 93 — Commands for all device types (part 1 of 2)**

Command name	Operation code	Type	Reference
ACCESS CONTROL IN	86h	O	8.3.2
ACCESS CONTROL OUT	87h	O	8.3.3
CHANGE ALIASES	A4h/0Bh <sup>a</sup>	O	6.2
EXTENDED COPY	83h	O	6.3
INQUIRY	12h	M	6.4
LOG SELECT	4Ch	O	6.5
LOG SENSE	4Dh	O	6.6
MANAGEMENT PROTOCOL IN	A3h/10h <sup>a</sup>	O	6.7
MANAGEMENT PROTOCOL OUT	A4h/10h <sup>a</sup>	O	6.8
MODE SELECT(6)	15h	C	6.9
MODE SELECT(10)	55h	C	6.10
MODE SENSE(6)	1Ah	C	6.11
MODE SENSE(10)	5Ah	C	6.12
PERSISTENT RESERVE IN	5Eh	C	6.13
PERSISTENT RESERVE OUT	5Fh	C	6.14
READ ATTRIBUTE	8Ch	O	6.15
READ BUFFER	3Ch	O	6.16
READ MEDIA SERIAL NUMBER	ABh/01h <sup>a</sup>	C	6.17
RECEIVE COPY RESULTS	84h	O	6.18
RECEIVE CREDENTIAL	7Fh/1800h <sup>a</sup>	O	6.19
RECEIVE DIAGNOSTIC RESULTS	1Ch	O	6.20
REPORT ALIASES	A3h/0Bh <sup>a</sup>	O	6.21
REPORT IDENTIFYING INFORMATION	A3h/05h <sup>a</sup>	O	6.22
REPORT LUNS	A0h	M	6.23
REPORT PRIORITY	A3h/0Eh <sup>a</sup>	O	6.24
REPORT SUPPORTED OPERATION CODES	A3h/0Ch <sup>a</sup>	O	6.25
REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS	A3h/0Dh <sup>a</sup>	O	6.26
REPORT TARGET PORT GROUPS	A3h/0Ah <sup>a</sup>	O	6.27
REPORT TIMESTAMP	A3h/0Fh <sup>a</sup>	O	6.28
REQUEST SENSE	03h	C	6.29
Type Key: C = Command implementation is defined in the applicable command standard (see 3.1.27). M = Command implementation is mandatory. O = Command implementation is optional. Z = Command implementation is defined in a previous standard.			
<sup>a</sup> This command is defined by a combination of operation code and service action. The operation code value is shown preceding the slash and the service action value is shown after the slash.			

Table 93 — Commands for all device types (part 2 of 2)

Command name	Operation code	Type	Reference
SECURITY PROTOCOL IN	A2h	O	6.30
SECURITY PROTOCOL OUT	B5h	O	6.31
SEND DIAGNOSTIC	1Dh	C	6.32
SET IDENTIFYING INFORMATION	A4h/06h <sup>a</sup>	O	6.33
SET PRIORITY	A4h/0Eh <sup>a</sup>	O	6.34
SET TARGET PORT GROUPS	A4h/0Ah <sup>a</sup>	O	6.35
SET TIMESTAMP	A4h/0Fh <sup>a</sup>	O	6.36
TEST UNIT READY	00h	M	6.37
WRITE ATTRIBUTE	8Dh	O	6.38
WRITE BUFFER	3Bh	C	6.39
Obsolete	16h	Z	
Obsolete	17h	Z	
Obsolete	18h	Z	
Obsolete	39h	Z	
Obsolete	3Ah	Z	
Obsolete	40h	Z	
Obsolete	56h	Z	
Obsolete	57h	Z	
Type Key: C = Command implementation is defined in the applicable command standard (see 3.1.27). M = Command implementation is mandatory. O = Command implementation is optional. Z = Command implementation is defined in a previous standard.			
<sup>a</sup> This command is defined by a combination of operation code and service action. The operation code value is shown preceding the slash and the service action value is shown after the slash.			

## 6.2 CHANGE ALIASES command

### 6.2.1 CHANGE ALIASES command introduction

The CHANGE ALIASES command (see table 94) requests that the device server maintain and make changes to a list of associations between eight byte alias values and SCSI target device or SCSI target port designations. A designation contains a name and optional identifier information that specifies a SCSI target device or SCSI target port (see 6.2.2). The alias list may be queried by the application client via the REPORT ALIASES command (see 6.2.1). If the REPORT ALIASES command is supported, the CHANGE ALIASES command shall also be supported.

The CHANGE ALIASES command is a service action of the MAINTENANCE OUT command. Additional MAINTENANCE OUT service actions are defined in SCC-2 and in this standard. The MAINTENANCE OUT service actions defined in SCC-2 apply only to logical units that return a device type of 0Ch or the sccs bit set to one in their standard INQUIRY data (see 6.4.2).

**Table 94 — CHANGE ALIASES command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A4h)							
1	Reserved			SERVICE ACTION (0Bh)				
2	Reserved							
5								
6	(MSB)	PARAMETER LIST LENGTH						(LSB)
9								
10	Reserved							
11	CONTROL							

The PARAMETER LIST LENGTH field specifies the length in bytes of the parameter data that shall be transferred from the application client to the device server. A parameter list length value of zero specifies that no data shall be transferred and no changes shall be made in the alias list.

If the parameter list length results in the truncation of the header or any alias entry, then the device server shall make no changes to the alias list and terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

On successful completion of a CHANGE ALIASES command, the device server shall maintain an association of each assigned eight byte alias value to the SCSI target device or SCSI target port designation. These associations shall be cleared by a logical unit reset or I\_T nexus loss. The device server shall maintain a separate alias list for each I\_T nexus.

A CHANGE ALIASES command may add, change or remove entries from the alias list. Alias list entries not referenced in the CHANGE ALIASES parameter data shall not be changed.

NOTE 18 - An application client may use alias values to reference SCSI target devices or SCSI target ports in third party commands (e.g., EXTENDED COPY). The alias list provides a mechanism for eight byte third party identifier fields to reference a third party device or port whose name or addressing information is longer than eight bytes. (E.g., an application may use the CHANGE ALIASES command to establish an association between an alias value and a SCSI target device or target port designation. Then, it may send an EXTENDED COPY command containing

in the parameter data an alias target descriptor (see 6.3.6.3) that includes this alias value. At the completion of the EXTENDED COPY command the application should clear this entry from the device server's alias list by sending a CHANGE ALIASES command that requests association of the alias value to a NULL DESIGNATION (see 6.2.4.2 alias format.)

If the device server has insufficient resources to make all requested changes to the alias list, then the device server shall make no changes to the alias list and shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INSUFFICIENT RESOURCES.

The parameter data for a CHANGE ALIASES command (see table 95) contains zero or more alias entries. If the device server processes a CHANGE ALIASES command that contains at least one alias entry while there exists any other enabled task that references an alias entry in the alias list, then the device server shall terminate the CHANGE ALIASES command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to OPERATION IN PROGRESS.

Table 95 — CHANGE ALIASES parameter list

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____							
3	PARAMETER DATA LENGTH (n-3)							(LSB)
4	Reserved _____							
7								
	Alias entry (or entries)							
8	Alias entry 0 (see 6.2.2) _____							
	⋮							
n	Alias entry x (see 6.2.2) _____							

The PARAMETER DATA LENGTH field should contain the number of bytes of attribute data and shall be ignored by the device server.

The format of an alias entry is described in 6.2.2.

### 6.2.2 Alias entry format

One alias entry (see table 96) describes one alias reported via the REPORT ALIASES command (see 6.21) or to be changed via the CHANGE ALIASES command.

**Table 96 — Alias entry format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
7	ALIAS VALUE (LSB)							
8	PROTOCOL IDENTIFIER							
9	Reserved							
10								
11	FORMAT CODE							
12	Reserved							
13								
14	(MSB)							
15	DESIGNATION LENGTH (n-15) (LSB)							
16	DESIGNATION							
n								

The ALIAS VALUE field contains the numeric alias value that the device server shall associate with the SCSI target device or target port specified by the values in the PROTOCOL IDENTIFIER, FORMAT CODE and DESIGNATION fields.

The PROTOCOL IDENTIFIER field (see table 97) specifies that the alias entry designation is independent of SCSI transport protocol or the SCSI transport protocol to which the alias entry applies.

**Table 97 — Alias entry PROTOCOL IDENTIFIER field**

Code	Description	Subclause
00h to 0Fh	Protocol specific designation	7.5.2
10h to 7Fh	Reserved	
80h	Protocol independent designation	6.2.4
81h to FFh	Reserved	

The FORMAT CODE field contents combined with the PROTOCOL IDENTIFIER field contents defines the format of the DESIGNATION field. The subclauses that describe each PROTOCOL IDENTIFIER field usage (see table 97) define the applicable FORMAT CODE field values.

The DESIGNATION LENGTH field specifies the number of bytes of the DESIGNATION field. The DESIGNATION LENGTH value shall be a multiple of four.

The zero-padded (see 4.4.2) DESIGNATION field should designate a unique SCSI target device or target port using the following:

- a) A SCSI device name or a target port name; and
- b) Optionally, one or more target port identifiers or SCSI transport protocol specific identifiers.



### 6.2.3 Alias designation validation

The device server shall not validate any designation at the time of processing either the REPORT ALIASES or CHANGE ALIASES command. Such validation shall occur only when the device server consults the alias list to resolve an alias to a designation in the context of third-party commands (e.g., EXTENDED COPY) or any other command that requires reference to the alias list.

If a designation identifies a unique SCSI target device or target port that is within a SCSI domain accessible to the device server, the designation is considered valid.

Based on the SCSI transport protocol specific requirements for a given designation format, a designation that does not identify a unique SCSI target device or target port within the SCSI domains accessible to the device server is considered invalid.

NOTE 19 - For example, a designation may be considered invalid if the device server has no ports on the SCSI domain of the designated SCSI target device or target port.

A designation having both SCSI name and SCSI port identifier information may be inconsistent if the device server is not able to access the named SCSI target device or target port through one or more of the names or identifiers in the designation. In such cases, the designation shall be treated as valid or invalid according to the SCSI transport protocol specific requirements.

#### Notes

- 20 For example, in FCP-2 both an N\_Port and World Wide Name for a SCSI port may be given in a designation. The designation definition may require that the N\_Port be that of the named port. In that case, the designation is invalid. Alternatively, the designation definition may view the N\_Port as a hint for the named FC Port accessible to the device server through a different D\_ID. In that case, the designation is valid and designate the named FC Port.
- 21 When only name information is provided in a designation, it is assumed that the device server has access to a mechanism for resolving names to identifiers. Access to such a service is SCSI transport protocol specific and vendor specific.

### 6.2.4 Alias entry protocol independent designations

#### 6.2.4.1 Alias entry protocol independent designations overview

The protocol independent alias entry designations have a protocol identifier of 80h and one of the format codes shown in table 98.

**Table 98 — Protocol independent alias entry format codes**

Format Code	Name	Designation Length (bytes)	Designation Contents	Subclause
00h	NULL DESIGNATION	0	none	6.2.4.2
01h to FFh	Reserved			

#### 6.2.4.2 NULL DESIGNATION alias format

In response to an alias entry with the NULL DESIGNATION format, the device server shall remove the specified alias value from the alias list. Application clients should use the NULL DESIGNATION format in a CHANGE ALIASES command to remove an alias entry from the alias list when that alias entry is no longer needed. The NULL DESIGNATION format shall not appear in REPORT ALIASES parameter data.

## 6.3 EXTENDED COPY command

### 6.3.1 EXTENDED COPY command introduction

The EXTENDED COPY command (see table 99) provides a means to copy data from one set of logical units to another set of logical units or to the same set of logical units. The entity within a SCSI device that receives and performs the EXTENDED COPY command is called the copy manager. The copy manager is responsible for copying data from the source devices to the destination devices. The copy source and destination devices are logical units that may reside in different SCSI devices or the same SCSI device. It is possible that the copy source device, copy destination device, and the copy manager are the same logical unit.

**Table 99 — EXTENDED COPY command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (83h)							
1	Reserved							
9								
10	(MSB)	PARAMETER LIST LENGTH						(LSB)
13								
14	Reserved							
15	CONTROL							

Before the copy manager is instructed to move data, the application controlling the data movement shall independently take any necessary actions required to prepare the copy source and destination devices for the EXTENDED COPY command (e.g. loading tapes, sending media changer commands, MODE SELECT commands, reservation commands, and/or tape positioning commands). After all preparatory actions have been accomplished, the EXTENDED COPY command should be issued to the copy manager to start the data transfer.

The PARAMETER LIST LENGTH field specifies the length in bytes of the parameter data that shall be contained in the Data-Out Buffer. A parameter list length of zero specifies that copy manager shall not transfer any data or alter any internal state; this shall not be considered an error. If the parameter list length causes truncation of the parameter list in a target descriptor or segment descriptor, then no data shall be transferred and the EXTENDED COPY command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

The EXTENDED COPY parameter list (see table 100) begins with a 16 byte header that contains the LIST IDENTIFIER field, the STR bit, the NRCR bit, the PRIORITY field, the length of the target descriptor list, the length of the segment descriptor list, and the length of the optional inline data. Immediately following the header is one or more target descriptors, followed by one or more segment descriptors, followed by any optional inline data.

**Table 100 — EXTENDED COPY parameter list**

Bit Byte	7	6	5	4	3	2	1	0	
0	LIST IDENTIFIER								
1	Reserved		STR	NRCR	Reserved	PRIORITY			
2	(MSB)	TARGET DESCRIPTOR LIST LENGTH (n-15)							(LSB)
3									
4	Reserved								
7									
8	(MSB)	SEGMENT DESCRIPTOR LIST LENGTH (m-n)							(LSB)
11									
12	(MSB)	INLINE DATA LENGTH (k-m)							(LSB)
15									
	Target descriptor(s)								
16	Target descriptor 0								
47									
	⋮								
n-31	Target descriptor x								
n									
	Segment descriptor(s)								
n+1	Segment descriptor 0 (See specific table for length.)								
n+1+l									
	⋮								
	Segment descriptor y (See specific table for length.)								
m									
m+1	Inline data								
k									

NOTE 22 - Unexpected results may occur when an application client fails to zero the reserved bytes in this parameter list. Copy managers should ensure that the reserved bytes 4 to 7 contain zeros.

The LIST IDENTIFIER field is a value selected by the application client to uniquely identify the extended copy operation to the copy manager. The list identifier also may be used in the RECEIVE COPY RESULTS command (see 6.18) to request status for a specific EXTENDED COPY command. The LIST IDENTIFIER value shall be unique for each concurrent EXTENDED COPY command sent via one I\_T nexus. If the copy manager detects a duplicate

LIST IDENTIFIER value, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to OPERATION IN PROGRESS.

The PRIORITY field establishes the priority of data transfer operations resulting from this EXTENDED COPY command relative to data transfer operations resulting from other commands being processed by the same device server. All commands other than copy commands have a priority of 1h. Priority 0h is the highest priority, with increasing PRIORITY values indicating lower priorities.

A Sequential Striped (STR) bit set to one specifies to the copy manager that the majority of the disk references in the parameter list represent sequential access of several striped disks. This may be used by the copy manager to perform read operations from a copy source disk at any time and in any order during processing of an EXTENDED COPY command as described in 6.3.6.4. A STR bit set to zero specifies to the copy manager that disk references may not be sequential.

If the No Receive Copy Results (NRCR) bit is set to zero, the copy manager shall hold data for retrieval by the application client using the RECEIVE COPY RESULTS command with the RECEIVE DATA service action (see 6.18.3) and specified by the segment descriptors. If NRCR is one, the copy manager may discard all data accessible to the application client via the RECEIVE COPY RESULTS command with the RECEIVE DATA service action. If the application client requests delivery of data that has been discarded as a result of NRCR being one, the copy manager shall respond as if the EXTENDED COPY command has not been processed.

The TARGET DESCRIPTOR LIST LENGTH contains the length in bytes of the target descriptor list that immediately follows the parameter list header. The number of target descriptors equals the length in bytes of the target descriptor list divided by 32.

An EXTENDED COPY command may reference one or more copy target devices (i.e., the name given by the EXTENDED COPY command description to copy source and/or the destination devices). Each copy target device is described by a target descriptor. All target descriptors have their formats specified by an EXTENDED COPY descriptor code. A copy manager may not support all target descriptor formats, however, the copy manager shall list all target descriptor formats supported in response to the RECEIVE COPY RESULTS command with OPERATING PARAMETERS service action (see 6.18.4). See 6.3.6 for a detailed description of the target descriptors.

Segment descriptors reference target descriptors by their position, or index, in the target descriptor list. The index for a target descriptor is computed by subtracting 16 from the starting byte number for the target descriptor in the parameter data and dividing the result by 32. The maximum number of target descriptors permitted within a parameter list is indicated by the MAXIMUM TARGET COUNT field in the copy manager's operating parameters (see 6.18.4). If the number of target descriptors exceeds the allowed number, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to TOO MANY TARGET DESCRIPTORS.

The SEGMENT DESCRIPTOR LIST LENGTH contains the length in bytes of the segment descriptor list that follows the target descriptors. See 6.3.7 for a detailed description of the segment descriptors. The maximum number of segment descriptors permitted within a parameter list is indicated by the MAXIMUM SEGMENT COUNT field in the copy manager's operating parameters (see 6.18.4). If the number of segment descriptors exceeds the allowed number, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to TOO MANY SEGMENT DESCRIPTORS.

The maximum length of the target and segment descriptors permitted within a parameter list is indicated by the MAXIMUM DESCRIPTOR LIST LENGTH field in the copy manager's operating parameters (see 6.18.4). If the combined length of the target and segment descriptors exceeds the allowed value, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

The **INLINE DATA LENGTH** field contains the number of bytes of inline data, after the last segment descriptor. A value of zero specifies that no inline data is present.

The copy manager shall move data from the copy source devices to the copy destination devices as specified by the segment descriptors. The specific commands issued by the copy manager to the copy source and destination devices while processing the segment descriptors is vendor specific. Upon completion of an **EXTENDED COPY** command with **GOOD** status, the copy source and destination devices, particularly stream devices, shall be positioned at deterministic locations such that the device may be repositioned to the same location by the application client with appropriate commands.

### 6.3.2 Errors detected before starting processing of the segment descriptors

Errors may occur during processing of an **EXTENDED COPY** command before the first segment descriptor is processed. These errors include CRC or parity errors while transferring the **EXTENDED COPY** command, invalid parameters in the CDB or parameter data, invalid segment descriptors, and inability of the copy manager to continue operating. In the event of such an exception condition, the copy manager shall:

- a) Terminate the **EXTENDED COPY** command with **CHECK CONDITION** status; and
- b) Set the **VALID** bit in the sense data to zero. The sense key shall contain the sense key code describing the exception condition (i.e., not **COPY ABORTED**).

### 6.3.3 Errors detected during processing of segment descriptors

Errors may occur after the copy manager has begun processing segment descriptors. These errors include invalid parameters in segment descriptors, invalid segment descriptors, unavailable targets referenced by target descriptors, inability of the copy manager to continue operating, and errors reported by source or destination copy target devices. If the copy manager receives **CHECK CONDITION** status from one of the copy target devices, it shall recover the sense data associated with the exception condition and clear any **ACA** condition associated with the **CHECK CONDITION** status.

If it is not possible to complete processing of a segment because the copy manager is unable to establish communications with a copy target device, because the copy target device does not respond to **INQUIRY**, or because the data returned in response to **INQUIRY** indicates an unsupported logical unit, then the **EXTENDED COPY** command shall be terminated with **CHECK CONDITION** status, with the sense key set to **COPY ABORTED**, and the additional sense code set to **COPY TARGET DEVICE NOT REACHABLE**.

If it is not possible to complete processing of a segment because the data returned in response to an **INQUIRY** command indicates a device type that does not match the type in the target descriptor, then the **EXTENDED COPY** command shall be terminated with **CHECK CONDITION** status, with the sense key set to **COPY ABORTED**, and the additional sense code set to **INCORRECT COPY TARGET DEVICE TYPE**.

If the copy manager has issued a command other than **INQUIRY** to a copy target device while processing an **EXTENDED COPY** command and the copy target device either fails to respond with status or responds with status other than **BUSY**, **TASK SET FULL**, **ACA ACTIVE**, or **RESERVATION CONFLICT**, then the condition shall be considered a copy target device command failure. In response to a copy target device command failure the **EXTENDED COPY** command shall be terminated with **CHECK CONDITION** status, with the sense key set to **COPY ABORTED**, and the additional sense code set to **THIRD PARTY DEVICE FAILURE**.

If a copy target device completes a command from the copy manager with a status of **BUSY**, **TASK SET FULL**, **ACA ACTIVE**, or **RESERVATION CONFLICT**, the copy manager shall either retry the command or terminate the **EXTENDED COPY** command as a copy target device command failure.

## NOTES

- 23 The copy manager is assumed to employ a vendor specific retry policy that minimizes time consuming and/or fruitless repetition of retries.
- 24 RESERVATION CONFLICT status is listed only to give the copy manager leeway in multi-port cases. The copy manager may have multiple initiator ports that are capable of reaching a copy target device, and a persistent reservation may restrict access to a single I\_T nexus. The copy manager may need to try access from multiple initiator ports to find the correct I\_T nexus.

If a copy target device responds to an input or output operation with a GOOD status but less data than expected is transferred, then the EXTENDED COPY command shall be terminated with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to COPY TARGET DEVICE DATA UNDERRUN. If an overrun is detected, then the EXTENDED COPY command shall be terminated with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to COPY TARGET DEVICE DATA OVERRUN.

Following an exception condition detected during segment descriptor processing:

- a) The copy manager shall terminate the EXTENDED COPY command with CHECK CONDITION status, with the sense key set to COPY ABORTED;
- b) The copy manager shall indicate the segment that was being processed at the time of the exception by writing the segment number to the third and fourth bytes of the COMMAND-SPECIFIC INFORMATION field. The segment number is based on the relative position of the segment descriptor in the EXTENDED COPY parameter list (i.e., the first segment descriptor in the parameter list is assigned descriptor number zero, the second is assigned one, etc.);
- c) If any data has been written to the destination for the segment being processed at the time the error occurred, the residual for the segment shall be placed in the INFORMATION field, and the VALID bit shall be set to one. The residual count shall be reported in bytes if the peripheral device type in the destination target descriptor is 03h (i.e., processor device), and in destination device blocks for all other device type codes. The residual count shall be computed by subtracting the number of bytes or blocks successfully written during the processing of the current segment from the number of bytes or blocks which would have been written if all commands had completed with GOOD status and all READ commands had returned the full data length requested. When computing the residual count, the copy manager shall include only the results of commands successfully completed by a copy destination device (i.e., commands completed by a copy destination device with GOOD status or with CHECK CONDITION status and the EOM bit set to one in the sense data). If the copy manager has used out of order transfers, the residual count shall be based only on the contiguous successfully completed transfers starting at relative byte zero of the segment (i.e., any successfully completed transfers farther from relative byte zero than the first incomplete or unsuccessful transfer shall not contribute to the computation of the residual count). If no data has been written to the destination for the segment being processed at the time the error occurred, then the VALID bit shall be set to zero and the contents of the INFORMATION field are not defined. Segment descriptors that do not specify a transfer count shall not have a valid residual count returned;
- d) If the exception condition is reported by the copy source device and fixed format sense data (see 4.5.3) is being returned, then the first byte of the COMMAND-SPECIFIC INFORMATION field shall be set to the starting byte number, relative to the first byte of sense data, of an area that contains the status byte and sense data delivered to the copy manager by the copy source device. The status byte and sense data shall not be modified by the copy manager or device server. A zero value indicates that no status byte and sense data is being returned for the copy source device;
- e) If the exception condition is reported by the copy destination device and fixed format sense data is being returned, then the second byte of the COMMAND-SPECIFIC INFORMATION field shall be set to the starting byte number, relative to the first byte of sense data, of an area that contains the status byte and sense data delivered to the copy manager by the copy destination device. The status byte and sense data shall not be modified by the copy manager or device server. A zero value indicates that no status byte and sense data is being returned for the copy destination device;

- f) If segment processing is terminated because a copy target device is unreachable or as the result of a failure in a command sent to a copy target device, then the SENSE-KEY SPECIFIC field shall be set as described in 4.5.2.4.5, with the FIELD POINTER field indicating the first byte of the target descriptor that identifies the copy target device;
- g) If, during the processing of a segment descriptor, the copy manager detects an error in the segment descriptor, then the SENSE-KEY SPECIFIC field shall be set as described in 4.5.2.4.5, with the FIELD POINTER field indicating the byte in error. The FIELD POINTER field may be used to indicate an offset into either the parameter data or the segment descriptor. The SD bit is used to differentiate between these two cases. The SD bit shall be set to zero to indicate the FIELD POINTER field contains an offset from the start of the parameter data. The SD bit shall be set to one to indicate the FIELD POINTER field contains an offset from the start of the segment descriptor; and
- h) The copy manager shall preserve information for the FAILED SEGMENT DETAILS service action of the RECEIVE COPY RESULTS command (see 6.18.5). The information shall be discarded as described in 6.18.5.

#### 6.3.4 Abort task management functions

When a device server processes an ABORT TASK, ABORT TASK SET, or CLEAR TASK SET task management function that terminates an EXTENDED COPY command, the copy manager shall ensure that all commands and data transfers generated by the terminated EXTENDED COPY command have been terminated and are no longer transferring data before allowing the task manager to complete the task management function. This requirement shall also apply to the processing the PREEMPT AND ABORT service action of the PERSISTENT RESERVE OUT command as described in 5.7.11.5.

### 6.3.5 Descriptor type codes

Target descriptors and segment descriptors share a single set of code values that identify the type of descriptor (see table 101). Segment descriptors use codes in the range 00h to BFh. The definitions of codes between C0h and DFh are vendor specific. Target descriptors use codes in the range E0h to FFh.

**Table 101 — EXTENDED COPY descriptor type codes (part 1 of 2)**

Descriptor type code	Reference	Description <sup>a</sup>	Shorthand <sup>a</sup>
00h	6.3.7.3	Copy from block device to stream device	block→ stream
01h	6.3.7.4	Copy from stream device to block device	stream→ block
02h	6.3.7.5	Copy from block device to block device	block→ block
03h	6.3.7.6	Copy from stream device to stream device	stream→ stream
04h	6.3.7.7	Copy inline data to stream device	inline→ stream
05h	6.3.7.8	Copy embedded data to stream device	embedded→ stream
06h	6.3.7.9	Read from stream device and discard	stream→ discard
07h	6.3.7.10	Verify block or stream device operation	
08h	6.3.7.11	Copy block device with offset to stream device	block<o>→ stream
09h	6.3.7.12	Copy stream device to block device with offset	stream→ block<o>
0Ah	6.3.7.13	Copy block device with offset to block device with offset	block<o>→ block<o>
0Bh	6.3.7.3	Copy from block device to stream device and hold a copy of processed data for the application client <sup>b</sup>	block→ stream +application client
0Ch	6.3.7.4	Copy from stream device to block device and hold a copy of processed data for the application client <sup>b</sup>	stream→ block +application client
0Dh	6.3.7.5	Copy from block device to block device and hold a copy of processed data for the application client <sup>b</sup>	block→ block +application client
0Eh	6.3.7.6	Copy from stream device to stream device and hold a copy of processed data for the application client <sup>b</sup>	stream→ stream +application client
0Fh	6.3.7.9	Read from stream device and hold a copy of processed data for the application client <sup>b</sup>	stream→ discard +application client
10h	6.3.7.14	Write filemarks to sequential-access device	filemark→ tape
11h	6.3.7.15	Space records or filemarks on sequential-access device	space→ tape
12h	6.3.7.16	Locate on sequential-access device	locate→ tape
<sup>a</sup> Block devices are those with peripheral device type codes 0h (i.e., direct access block), 5h (i.e., CD/DVD), and Eh (i.e., simplified direct-access). Stream devices are those devices with peripheral device type codes 1h (i.e., sequential-access) and 3h (i.e., processor). Sequential-access stream (indicated by the term tape in the shorthand column) devices are those with peripheral device type code 1h. See 6.4.2 for peripheral device type code definitions.			
<sup>b</sup> The application client shall use the RECEIVE COPY RESULTS with a RECEIVE DATA service action to retrieve data held for it by the copy manager (see 6.18.3).			



**Table 101 — EXTENDED COPY descriptor type codes (part 2 of 2)**

<b>Descriptor type code</b>	<b>Reference</b>	<b>Description <sup>a</sup></b>	<b>Shorthand <sup>a</sup></b>
13h	6.3.7.17	Image copy from sequential-access device to sequential-access device	<i>tape→ <i>tape
14h	6.3.7.18	Register persistent reservation key	
15h	6.3.7.19	Third party persistent reservations source I_T nexus	
16h to BFh		Reserved for segment descriptors	
C0h to DFh		Vendor specific descriptors	
E0h	7.5.3.2	Fibre Channel N_Port_Name target descriptor	
E1h	7.5.3.3	Fibre Channel N_Port_ID target descriptor	
E2h	7.5.3.4	Fibre Channel N_Port_ID with N_Port_Name checking target descriptor	
E3h	7.5.3.5	Parallel Interface T_L target descriptor	
E4h	6.3.6.2	Identification descriptor target descriptor	
E5h	7.5.3.8	IPv4 target descriptor	
E6h	6.3.6.3	Alias target descriptor	
E7h	7.5.3.7	RDMA target descriptor	
E8h	7.5.3.6	IEEE 1394 EUI-64 target descriptor	
E9h	7.5.3.9	SAS Serial SCSI Protocol target descriptor	
EAh to FFh		Reserved for target descriptors	
<sup>a</sup> Block devices are those with peripheral device type codes 0h (i.e, direct access block), 5h (i.e., CD/DVD), and Eh (i.e., simplified direct-access). Stream devices are those devices with peripheral device type codes 1h (i.e., sequential-access) and 3h (i.e., processor). Sequential-access stream (indicated by the term tape in the shorthand column) devices are those with peripheral device type code 1h. See 6.4.2 for peripheral device type code definitions. <sup>b</sup> The application client shall use the RECEIVE COPY RESULTS with a RECEIVE DATA service action to retrieve data held for it by the copy manager (see 6.18.3).			

## 6.3.6 Target descriptors

### 6.3.6.1 Target descriptors introduction

All target descriptors (see table 102) are 32 bytes in length and begin with a four-byte header containing the DESCRIPTOR TYPE CODE field that identifies the format of the descriptor. The assigned descriptor type code values are shown in table 101. Support for each target descriptor format is optional. If a copy manager receives an unsupported descriptor type code in a target descriptor, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to UNSUPPORTED TARGET DESCRIPTOR TYPE CODE.

**Table 102 — Target descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E0h to FFh)							
1	LU ID TYPE		NUL	PERIPHERAL DEVICE TYPE				
2	(MSB)							
3	RELATIVE INITIATOR PORT IDENTIFIER							(LSB)
4	Target descriptor parameters							
27								
28								
31	Device type specific parameters							

The DESCRIPTOR TYPE CODE field is described in 6.3.5.

The LU ID TYPE field (see table 103) specifies the interpretation of the LU IDENTIFIER field in target descriptors that contain a LU IDENTIFIER field.

**Table 103 — LU ID TYPE field**

Code	LU IDENTIFIER field contents	Reference
00b	Logical Unit Number	SAM-4
01b	Proxy Token	8.3.1.6.2
10b to 11b	Reserved	

Support for LU ID type codes other than 00b is optional. If a copy manager receives an unsupported LU ID type code, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the LU ID TYPE field specifies that the LU IDENTIFIER field contains a logical unit number, then the LU IDENTIFIER field specifies the logical unit within the SCSI device specified by other fields in the target descriptor that shall be the source or destination for EXTENDED COPY operations.

If the LU ID TYPE field specifies that the LU IDENTIFIER field contains a proxy token (see 8.3.1.6.2), then the copy manager shall use the LU IDENTIFIER field contents to obtain proxy access rights to the logical unit associated with the proxy token. The logical unit number that represents the proxy access rights shall be the source or destination for EXTENDED COPY operations.

The copy manager should obtain a LUN value for addressing this logical unit by sending an ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action (see 8.3.3.11) to the access controls coordinator of the SCSI target device that is identified by other fields in the target descriptor. The copy manager shall use a LUN assigned on the basis of a proxy token only for those commands that are necessary for the processing of the EXTENDED COPY command whose parameter data contains the proxy token. When the copy manager has completed EXTENDED COPY commands involving a proxy token, the copy manager should release the LUN value using an ACCESS CONTROL OUT command with RELEASE PROXY LUN service action (see 8.3.3.12).

EXTENDED COPY access to proxy logical units is to be accomplished only via LU ID type 01b. If the copy manager receives a target descriptor containing LU ID type 00b and a logical unit number matching a LUN value that the copy manager has obtained using an ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action, then the EXTENDED COPY command shall be terminated with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to COPY TARGET DEVICE NOT REACHABLE.

A null device (NUL) bit set to zero specifies that the target descriptor identifies a SCSI device that is expected to respond to an INQUIRY command and to which data movement commands may be sent. A NUL bit set to one specifies that the descriptor identifies a null device that is not expected to be the recipient of any SCSI commands. If NUL is one, bytes 4 to 27 of the target descriptor shall be ignored. If the processing required by a segment descriptor necessitates sending a command to a copy target device whose target descriptor has the NUL bit set to one, then the EXTENDED COPY command shall be terminated as if an unreachable copy target device had been encountered (see 6.3.3).

NOTE 25 - Target descriptors with the NUL bit set to one are useful for processing the residual data from previous segment descriptors without affecting any media. (E.g., a segment descriptor of type 06h (stream device to discard) with a byte count of zero, CAT equal to zero, and a null source target descriptor with PAD equal to one may be used to discard all residual data.)

The PERIPHERAL DEVICE TYPE field is described in 6.4.2. The value in the DESCRIPTOR TYPE CODE field determines the format of the target descriptor parameters that follow the four-byte header and precede the device type specific parameters. The values in the DESCRIPTOR TYPE CODE field are listed in table 101.

The value in the PERIPHERAL DEVICE TYPE field determines the format of the device type specific parameters that follow the target descriptor parameters. The device type specific parameters convey information specific to the type of device identified by the target descriptor.

Table 104 lists the peripheral device type code values having formats defined for the device type specific parameters in a target descriptor. Peripheral device types with code values not listed in table 104 are reserved in the EXTENDED COPY parameter list.

**Table 104 — Device type specific parameters in target descriptors**

Peripheral Device Type	Reference	Description	Shorthand
00h, 04h, 05h, 07h, and 0Eh	6.3.6.4	Block devices	Block
01h	6.3.6.5	Sequential-access devices	Stream or Tape
03h	6.3.6.6	Processor devices	Stream

The RELATIVE INITIATOR PORT IDENTIFIER field specifies the relative port identifier (see 3.1.120) of the initiator port within the SCSI device that the copy manager shall use to access the logical unit described by the target descriptor, if such access requires use of an initiator port (i.e., if the logical unit is in the same SCSI device as the copy manager, the RELATIVE INITIATOR PORT IDENTIFIER field is ignored). A RELATIVE INITIATOR PORT IDENTIFIER field set to zero specifies that the copy manager may use any initiator port or ports within the SCSI device.

The copy manager may, as part of processing a segment descriptor, verify the information in a target descriptor's device specific fields. However, when verifying the information, the copy manager shall not issue any commands that change the position of read/write media on the copy target device without returning the media to its original position. Any errors encountered while verifying the information shall be handled as described in 6.3.3.

### 6.3.6.2 Identification descriptor target descriptor format

The target descriptor format shown in table 105 instructs the copy manager to locate a SCSI target device and logical unit that returns a Device Identification VPD page (see 7.7.3) containing an Identification descriptor having the specified CODE SET, ASSOCIATION, DESIGNATOR TYPE, IDENTIFIER LENGTH, and IDENTIFIER field values. The copy manager may use any N\_Port, target port identifier and logical unit number values that result in matching VPD field values to address the logical unit. If multiple target port identifiers and logical unit number combinations access matching VPD field values, the copy manager may use any combination to address the logical unit and shall try other combinations in the event that one combination becomes non-operational during the processing of an EXTENDED COPY command.

**Table 105 — Identification descriptor target descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E4h)							
1	LU ID TYPE		NUL	PERIPHERAL DEVICE TYPE				
2	(MSB) _____							
3	RELATIVE INITIATOR PORT IDENTIFIER _____ (LSB)							
4	Reserved				CODE SET			
5	Reserved		ASSOCIATION		DESIGNATOR TYPE			
6	Reserved							
7	DESIGNATOR LENGTH (n-7)							
8	DESIGNATOR _____							
n								
n+1	Reserved _____							
27								
28	Device type specific parameters _____							
31								

The DESCRIPTOR TYPE CODE field, PERIPHERAL DEVICE TYPE field, NUL bit, RELATIVE INITIATOR PORT IDENTIFIER field, and the device type specific parameters are described in 6.3.6.1.

The LU ID TYPE field is reserved for this target descriptor.

The CODE SET field contains a code set enumeration (see 4.4.3) that indicates the format of the DESIGNATOR field.

The contents of the ASSOCIATION, DESIGNATOR TYPE, DESIGNATOR LENGTH, and DESIGNATOR fields are specified in 7.7.3.

The designator length shall be 20 or less. If the designator length is 20, there shall be no reserved bytes between the target descriptor parameters and the device type specific parameters.

Some combinations of code set, association, designator type, designator length and designator do not uniquely identify a logical unit to serve as a copy target device. The behavior of the copy manager when such combinations are received is unpredictable.

### 6.3.6.3 Alias target descriptor format

The target descriptor format shown in table 106 instructs the copy manager to locate a SCSI target port and logical unit using the alias list (see 3.1.8) designation associated with the specified alias value. The alias list is maintained using the CHANGE ALIASES command (see 6.2).

**Table 106 — Alias target descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E6h)							
1	LU ID TYPE		NUL	PERIPHERAL DEVICE TYPE				
2	(MSB)							
3	RELATIVE INITIATOR PORT IDENTIFIER							(LSB)
4								
11	LU IDENTIFIER							
12								
19	ALIAS VALUE							
20								
27	Reserved							
28								
31	Device type specific parameters							

The DESCRIPTOR TYPE CODE field, PERIPHERAL DEVICE TYPE field, NUL bit, RELATIVE INITIATOR PORT IDENTIFIER field, and the device type specific parameters are described in 6.3.6.1.

The ALIAS VALUE field specifies an alias value in the alias list as managed by the CHANGE ALIASES command (see 6.2) and maintained by the device server.

When the device server first processes an alias target descriptor, it shall check the value of the ALIAS VALUE field for a corresponding entry in the alias list. If the value is not in the alias list or the device server is unable to validate the designation (see 6.2.3) associated with the alias value, the command shall be terminated because the copy target device is unavailable (see 6.3.3). An application client generating EXTENDED COPY commands that include alias target descriptors in the parameter data is responsible for providing a valid entry in the alias list using the CHANGE ALIASES command (see 6.2) prior to sending the EXTENDED COPY command.

### 6.3.6.4 Device type specific target descriptor parameters for block device types

The format for the device type specific target descriptor parameters for block device types (i.e., device type code values 00h, 04h, 05h, 07h, and 0Eh) is shown in table 107.

**Table 107 — Device type specific target descriptor parameters for block device types**

Bit Byte	7	6	5	4	3	2	1	0
28	Reserved					PAD	Reserved	
29	(MSB)							
30	DISK BLOCK LENGTH							
31	(LSB)							

The PAD bit is used in conjunction with the CAT bit (see 6.3.7.1) in the segment descriptor to determine what action should be taken when a segment of the copy does not fit exactly into an integer number of destination blocks (see 6.3.7.2).

The DISK BLOCK LENGTH field contains the number of bytes in a disk block, excluding any protection information (see SBC-2), for the logical device being addressed.

The copy manager may read ahead from sources of block device type (i.e., the copy manager may perform read operations from a source disk at any time and in any order during processing of an EXTENDED COPY command, provided that the relative order of writes and reads on the same blocks within the same target descriptor does not differ from their order in the segment descriptor list).

### 6.3.6.5 Device type specific target descriptor parameters for sequential-access device types

The format for the device type specific target descriptor parameters for the sequential-access device type (i.e., device type code value 01h) is shown in table 108.

**Table 108 — Device type specific target descriptor parameters for sequential-access device types**

Bit Byte	7	6	5	4	3	2	1	0
28	Reserved					PAD	Reserved	FIXED
29	(MSB)							
30	STREAM BLOCK LENGTH							
31	(LSB)							

The contents of the FIXED bit and STREAM BLOCK LENGTH field are combined with the STREAM DEVICE TRANSFER LENGTH FIELD in the segment descriptor to determine the length of the stream read or write operation as specified in table 109.

**Table 109 — Stream device transfer lengths**

<b>FIXED bit</b>	<b>STREAM BLOCK LENGTH field</b>	<b>Description</b>
0	000000h	Use variable length reads or writes. The number of bytes for each read or write is specified by the STREAM DEVICE TRANSFER LENGTH field in the segment descriptor.
0	000001h to FFFFFFFh	The command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.
1	000000h	The command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.
1	000001h to FFFFFFFh	Use fixed record length reads or writes. The number of bytes for each read or write shall be the product of the STREAM BLOCK LENGTH field and the STREAM DEVICE TRANSFER LENGTH field in the segment descriptor.

The PAD bit is used in conjunction with the CAT bit (see 6.3.7.2) in the segment descriptor to determine what action should be taken when a segment of the copy does not fit exactly into an integer number of destination blocks (see 6.3.7.2).

All read commands issued to sequential-access type devices shall have the SILI bit equal to zero.

The copy manager shall not read ahead from sources of stream device type (i.e., the read operations required by a segment descriptor for which the source is a stream device shall not be started until all write operations for previous segment descriptors have completed).

#### 6.3.6.6 Device type specific target descriptor parameters for processor device types

The format for the device type specific target descriptor parameters for the processor device type (i.e., device type code value 03h) is shown in table 110.

**Table 110 — Device type specific target descriptor parameters for processor device types**

Bit Byte	7	6	5	4	3	2	1	0
28	Reserved					PAD	Reserved	
29	Reserved							
31								

The PAD bit is used in conjunction with the CAT bit (see 6.3.7.2) in the segment descriptor to determine what action should be taken when a segment of the copy does not fit exactly into an integer number of SEND or RECEIVE commands (see 6.3.7.2).

When the processor device is a source, the number of bytes to be transferred by a SEND command shall be specified by STREAM DEVICE TRANSFER LENGTH field in the segment descriptor. When the processor device is a destination, the number of bytes to be transferred by a RECEIVE command shall be specified by STREAM DEVICE TRANSFER LENGTH field in the segment descriptor.

### 6.3.7 Segment descriptors

#### 6.3.7.1 Segment descriptors introduction

Segment descriptors (see table 111) begin with an eight byte header.

**Table 111 — Segment descriptor header**

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (00h-3Fh)							
1	Reserved						DC	CAT
2	(MSB)	DESCRIPTOR LENGTH (n-7)						
3								
4	(MSB)	SOURCE TARGET DESCRIPTOR INDEX						
5								
6	(MSB)	DESTINATION TARGET DESCRIPTOR INDEX						
7								
8	Segment descriptor parameters							
n								

The DESCRIPTOR TYPE CODE field is described in 6.3.5. Support for each segment descriptor format is optional. If a copy manager receives an unsupported descriptor type code in a segment descriptor, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to UNSUPPORTED SEGMENT DESCRIPTOR TYPE CODE.

The destination count (DC) bit is only applicable to segment descriptors with descriptor type code values of 02h and 0Dh (see 6.3.7.5). The DC bit is reserved for all other segment descriptors.

The CAT bit is described in 6.3.7.2.

The DESCRIPTOR LENGTH field contains the length in bytes of the fields that follow the DESCRIPTOR LENGTH field in the segment descriptor. For each descriptor type code value, the length should be constant.

The SOURCE TARGET DESCRIPTOR INDEX field contains an index into the target descriptor list (see 6.3.1) identifying the source copy target device. The DESTINATION TARGET DESCRIPTOR INDEX field contains an index into the target descriptor list (see 6.3.1) identifying the destination copy target device. Some segment descriptor formats do not require a SOURCE TARGET DESCRIPTOR INDEX field or a DESTINATION TARGET DESCRIPTOR INDEX field, in which case the field is reserved.

If the copy target device identified by a SOURCE TARGET DESCRIPTOR INDEX field or a DESTINATION TARGET DESCRIPTOR INDEX field is not accessible to the copy manager, then the command shall be terminated with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to UNREACHABLE COPY TARGET.



### 6.3.7.2 Segment descriptor processing

In processing a segment descriptor, the copy manager may be required to:

- a) Read source data by issuing data input commands to the source device;
- b) Process data, an operation that generally designates data as destination data intended for transfer to the destination device; and
- c) Write some or all of the destination data to the destination device.

The number of blocks to read and write, the number of bytes to process, and the nature of processing are determined by the segment descriptor type code, the parameters of the segment descriptor, and the amount of residual source or destination data retained from the previous segment, if any.

Except as otherwise specified by particular segment descriptor type codes:

- a) Just enough whole-block read operations shall be performed to supply, together with residual source data from the previous segment or segments, the number of bytes to be processed;
- b) Processing consists of removing bytes from the source data and designating them as destination data, without change; and
- c) As many whole-block write operations as possible shall be performed with the destination data, including any residual destination data from the previous segment or segments.

Any residual source data from the previous segment or segments shall be processed before any data read from the source device during processing of the current segment descriptor. Any residual destination data from the previous segment or segments shall be written before any data processed during processing of the current segment descriptor.

Exceptions and clarifications to these general rules are described in table 112 and the subclauses it references.

**Table 112 — Descriptor Type Code Dependent Copy Manager Processing (part 1 of 2)**

Segment Descriptor Type Code	Reference	Description
00h (block→ stream) or 0Bh (block→ stream+application client)	6.3.7.3	The number of bytes processed is determined by the BLOCK DEVICE NUMBER OF BLOCKS field for the source blocks (see applicable type code definition subclauses for details). <sup>a</sup>
02h (block→ block) or 0Dh (block→ block+application client) with DC=0	6.3.7.5	
02h (block→ block) or 0Dh (block→ block+application client) with DC=1	6.3.7.5	The number of blocks or byte range specified shall be output to the destination device. If residual destination data is sufficient to perform the output, then no data shall be processed. Otherwise, just as much data as needed shall be processed (which may involve reading data from the source device) so that the destination data (which includes any residual destination data from the previous segment) is sufficient. <sup>a</sup>
01h (stream→ block) or 0Ch (stream→ block+application client)	6.3.7.3	
09h (stream→ block<o>)	6.3.7.12	
03h (stream→ stream) or 0Eh (stream→ stream+application client)	6.3.7.6	The number of bytes specified in the segment descriptor shall be processed. <sup>a</sup>
<sup>a</sup> For segment descriptor type codes 0Bh, 0Ch, 0Dh and 0Eh, a copy of the processed data shall also be held for retrieval by the application client.		

**Table 112 — Descriptor Type Code Dependent Copy Manager Processing (part 2 of 2)**

Segment Descriptor Type Code	Reference	Description
04h (inline→ stream)	6.3.7.7	The specified number of bytes of inline or embedded data shall be appended to the destination data, and no source data shall be processed.
05h (embedded→ stream)	6.3.7.8	
06h (stream→ discard)	6.3.7.9	The specified number of bytes shall be removed from the source data and discarded.
07h (verify device operation)	6.3.7.10	No data shall be processed and no read or write operations shall be performed on copy target devices. Residual source or destination data, if any, shall be retained or discarded as if the CAT bit were equal to one.
10h (filemark→ tape)	6.3.7.14	
11h (space→ tape)	6.3.7.15	
12h (locate→ tape)	6.3.7.16	
14h (register persistent reservation key)	6.3.7.18	
08h (block<o>→ stream)	6.3.7.11	The required blocks shall be read from the source device, the designated byte range shall be extracted as source data, and the designated number of bytes (starting with residual source data, if any) shall be processed.
0Ah (block<o>→ block<o>)	6.3.7.13	The source byte range specified shall be read into source data, the number of bytes specified shall be moved from source data to destination data, and the specified destination byte range shall be written using destination data.
0Fh (stream→ discard+application client)	6.3.7.9	The specified number of bytes shall be removed from the source data and held for retrieval by the application client.
13h (<i>tape→ <i>tape)	6.3.7.17	The data movement shall not involve processing as described in this subclause. Residual source or destination data, if any, shall not be used and shall be retained or discarded as if the CAT bit were equal to one.
<sup>a</sup> For segment descriptor type codes 0Bh, 0Ch, 0Dh and 0Eh, a copy of the processed data shall also be held for retrieval by the application client.		

Reads and writes shall be performed using whole-block transfer lengths determined by the block size, transfer length, or both. Therefore some source data may remain unprocessed and some destination data may not have been transferred at the end of a segment. If so, the residue shall be handled according to the CAT bit in the segment descriptor and the PAD bits of the source and destination target descriptors, as defined in table 113.

Table 113 — PAD and CAT bit definitions

PAD bit in		CAT bit	Copy manager action
Source target descriptor	Destination target descriptor		
0 or 1	0 or 1	1	Any residual source data shall be retained as source data for a subsequent segment descriptor. Any residual destination data shall be retained as destination data for a subsequent segment descriptor. It shall not be an error if either the source or destination target index in the following segment descriptor does not match the corresponding target index with which residual data was originally associated. If the CAT bit is set to one on the last segment of an EXTENDED COPY command, any residual data shall be discarded and this shall not be considered an error.
1	1	0	Any residual source data shall be discarded. Any residual destination data shall be padded with zeroes to make a whole block transfer. <sup>a</sup>
0	1	0	Any residual source data shall be handled as if the CAT bit is equal to one (i.e., discarded on the last segment and retained otherwise). Any residual destination data shall be padded with zeroes to make a whole block transfer. <sup>a</sup>
1	0	0	Any residual source or destination data shall be discarded.
0	0	0	If there is residual source or destination data, the EXTENDED COPY command shall be terminated with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to UNEXPECTED INEXACT SEGMENT.
<sup>a</sup> When the CAT bit is set to zero and the destination target descriptor has the PAD bit set to one, the EXTENDED COPY command shall be terminated with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to UNEXPECTED INEXACT SEGMENT if any of the following conditions are met: <ul style="list-style-type: none"> <li>a) If any residual destination data is present after writing the designated byte range for a segment descriptor of type 09h (stream→ block &lt;o&gt;) or 0Ah (block&lt;o&gt;→ block&lt;o&gt;); or</li> <li>b) If any residual destination data is present after the designated number of blocks have been written for a segment descriptor of type 02h (block→ block) with DC set to one, 0Dh (block→ block+application client) with DC set to one, 01h (stream→ block) or 0Ch (stream→ block+application client).</li> </ul>			

A few segment descriptors have either no source or no destination and handling of the PAD bit for those descriptors shall be as follows. For segment descriptor types 04h (inline→ stream, see 6.3.7.7) and 05h (embedded→ stream, see 6.3.7.8), the handling shall be as if the PAD is set to zero for the source target descriptor. For segment descriptor types 06h and 0Fh (stream→ discard and stream→ discard+application client, see 6.3.7.9), handling shall be as if the PAD is set to zero for the destination target descriptor.

### 6.3.7.3 Block device to stream device operations

The segment descriptor format shown in table 114 is used by the copy operations that move data from a block device to a stream device or vice versa.

**Table 114 — Block device to or from stream device segment descriptor**

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (00h, 01h, 0Bh, or 0Ch)							
1	Reserved							CAT
2	(MSB)	DESCRIPTOR LENGTH (0014h)						(LSB)
3								
4	(MSB)	SOURCE TARGET DESCRIPTOR INDEX						(LSB)
5								
6	(MSB)	DESTINATION TARGET DESCRIPTOR INDEX						(LSB)
7								
8	Reserved							
9	(MSB)	STREAM DEVICE TRANSFER LENGTH						(LSB)
10								
11								(LSB)
12	Reserved							
13	Reserved							
14	(MSB)	BLOCK DEVICE NUMBER OF BLOCKS						(LSB)
15								
16	(MSB)	BLOCK DEVICE LOGICAL BLOCK ADDRESS						(LSB)
23								

The DESCRIPTOR TYPE CODE field is described in 6.3.5 and 6.3.7.1. Two DESCRIPTOR TYPE CODE values use the segment descriptor format shown in table 114 and described in this subclause.

For descriptor type code 00h (block→ stream) or descriptor type code 0Bh (block→ stream+application client), the copy manager shall copy the data from the source block device identified by the SOURCE TARGET DESCRIPTOR INDEX field to the destination stream device identified by the DESTINATION TARGET DESCRIPTOR INDEX field using the logical blocks starting at the location identified by the BLOCK DEVICE LOGICAL BLOCK ADDRESS field. As many blocks shall be read as necessary to process (see 6.3.7.2) a number of bytes equal to the contents of the DISK BLOCK LENGTH field in the target descriptor for the source device times the contents of the BLOCK DEVICE NUMBER OF BLOCKS field. The data shall be written to the stream device starting at the current position of the media.

For descriptor type code 0Bh (block→ stream+application client), the copy manager also shall hold a copy of the processed data for delivery to the application client upon completion of the EXTENDED COPY command in response to a RECEIVE COPY RESULTS command with RECEIVE DATA service action as described in 6.18.3. The minimum amount of held data supported by the copy manager is returned in the response data for the RECEIVE COPY RESULTS command with OPERATING PARAMETERS service action (see 6.18.4). If the copy manager supports the 0Bh descriptor type code, it also shall support the RECEIVE COPY RESULTS command with RECEIVE DATA service action.

The CAT bit is described in 6.3.7.2.

The DESCRIPTOR LENGTH field shall contain 20 (0014h). The SOURCE TARGET DESCRIPTOR INDEX and DESTINATION TARGET DESCRIPTOR INDEX fields are described in 6.3.7.1.

The STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be written on each write operation to the stream device. See 6.3.6.5 for a description of how data in the STREAM DEVICE TRANSFER LENGTH field in the segment descriptor interacts with data in the STREAM BLOCK LENGTH field in the device type specific target descriptor parameters for the sequential-access device type.

The BLOCK DEVICE NUMBER OF BLOCKS field specifies the length, in source logical blocks, of data to be processed (see 6.3.7.2) in the segment. A value of zero shall not be considered as an error. No data shall be processed, but any residual destination data retained from a previous segment shall be written if possible to the destination in whole-block transfers. A value of zero shall not modify the handling of residual data.

The BLOCK DEVICE LOGICAL BLOCK ADDRESS field specifies the starting logical block address on the block device for this segment.

#### **6.3.7.4 Stream device to block device operations**

The segment descriptor format shown in table 114 (see 6.3.7.3) also is used by the copy operations that move data from a stream device to a block device. Two DESCRIPTOR TYPE CODE values use the segment descriptor format shown in table 114 and described in this subclause.

For descriptor type code 01h (stream→ block) or descriptor type code 0Ch (stream→ block+application client), the copy manager shall copy the data from the source stream device identified by the SOURCE TARGET DESCRIPTOR INDEX field to the destination block device identified by the DESTINATION TARGET DESCRIPTOR INDEX field using the stream data starting at the current position of the stream device. The data shall be written to logical blocks starting at the location identified by the BLOCK DEVICE LOGICAL BLOCK ADDRESS field and continuing for the number of blocks specified in the BLOCK DEVICE NUMBER OF BLOCKS field.

For descriptor type code 0Ch (stream→ block+application client), the copy manager also shall hold a copy of the processed data for delivery to the application client upon completion of the EXTENDED COPY command in response to a RECEIVE COPY RESULTS command with RECEIVE DATA service action as described in 6.18.3. The minimum amount of held data supported by the copy manager is returned in the response data for the RECEIVE COPY RESULTS command with OPERATING PARAMETERS service action (see 6.18.4). If the copy manager supports the 0Ch descriptor type code, it also shall support the RECEIVE COPY RESULTS command with RECEIVE DATA service action.

The CAT bit is described in 6.3.7.2.

The DESCRIPTOR LENGTH field shall contain 20 (0014h). The SOURCE TARGET DESCRIPTOR INDEX and DESTINATION TARGET DESCRIPTOR INDEX fields are described in 6.3.7.1.

The STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be read from the source stream device on each read operation. See 6.3.6.5 for a description of how data in the STREAM DEVICE TRANSFER LENGTH field in the segment descriptor interacts with data in the STREAM BLOCK LENGTH field in the device type specific target descriptor parameters for the sequential-access device type.

The BLOCK DEVICE NUMBER OF BLOCKS field specifies the number blocks to be written by the segment. A value of zero specifies that no blocks shall be written in this segment. This shall not be considered as an error.

The BLOCK DEVICE LOGICAL BLOCK ADDRESS field specifies the starting logical block address on the block device for this segment.

### 6.3.7.5 Block device to block device operations

The segment descriptor format shown in table 115 is used by the copy operations that move data from a block device to a block device.

**Table 115 — Block device to block device segment descriptor**

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (02h or 0Dh)							
1	Reserved						DC	CAT
2	(MSB)	DESCRIPTOR LENGTH (0018h)						
3								
4	(MSB)	SOURCE TARGET DESCRIPTOR INDEX						
5								
6	(MSB)	DESTINATION TARGET DESCRIPTOR INDEX						
7								
8	Reserved							
9	Reserved							
10	(MSB)	BLOCK DEVICE NUMBER OF BLOCKS						
11								
12	(MSB)	SOURCE BLOCK DEVICE LOGICAL BLOCK ADDRESS						
19								
20	(MSB)	DESTINATION BLOCK DEVICE LOGICAL BLOCK ADDRESS						
27								

The DESCRIPTOR TYPE CODE field is described in 6.3.5 and 6.3.7.1. Two DESCRIPTOR TYPE CODE values use the segment descriptor format shown in table 115 and described in this subclause.

For descriptor type code 02h (block→ block) or descriptor type code 0Dh (block→ block+application client), the copy manager shall copy the data from the source block device identified by the SOURCE TARGET DESCRIPTOR INDEX field to the destination block device identified by the DESTINATION TARGET DESCRIPTOR INDEX field using the logical blocks starting at the location identified by the SOURCE BLOCK DEVICE LOGICAL BLOCK ADDRESS field. The data shall be written to logical blocks starting at the location identified by the DESTINATION BLOCK DEVICE LOGICAL BLOCK ADDRESS field.

If the DC bit equals zero, then as many blocks shall be read as necessary to process (see 6.3.7.2) a number of bytes equal to the contents of the DISK BLOCK LENGTH field in the target descriptor for the source device times the contents of the BLOCK DEVICE NUMBER OF BLOCKS field, and as many writes as possible shall be performed using any residual destination data from the previous segment and the data processed in this segment. If the DC bit equals one, then the number of blocks specified by the BLOCK DEVICE NUMBER OF BLOCKS field shall be written to the destination block device, as many bytes shall be processed as necessary for these writes to be performed, and as many blocks shall be read as necessary to supply the data to be processed.

For descriptor type code 0Dh (block→ block+application client), the copy manager also shall hold a copy of the processed data for delivery to the application client upon completion of the EXTENDED COPY command in response to a RECEIVE COPY RESULTS command with RECEIVE DATA service action as described in 6.18.3.

The minimum amount of held data supported by the copy manager is returned in the response data for the RECEIVE COPY RESULTS command with OPERATING PARAMETERS service action (see 6.18.4). If the copy manager supports the 0Dh descriptor type code, it also shall support the RECEIVE COPY RESULTS command with RECEIVE DATA service action.

The CAT bit is described in 6.3.7.2.

The destination count (DC) bit specifies whether the BLOCK DEVICE NUMBER OF BLOCKS field refers to the source or destination device. A DC bit set to zero specifies that the BLOCK DEVICE NUMBER OF BLOCKS field refers to the source device. A DC bit set to one specifies that the BLOCK DEVICE NUMBER OF BLOCKS field refers to the destination device.

The DESCRIPTOR LENGTH field shall contain 24 (0018h). The SOURCE TARGET DESCRIPTOR INDEX and DESTINATION TARGET DESCRIPTOR INDEX fields are described in 6.3.7.1.

If DC is set to zero, the BLOCK DEVICE NUMBER OF BLOCKS field specifies the number of blocks to be processed. If DC is set to one, the BLOCK DEVICE NUMBER OF BLOCKS field specifies the number of blocks to be written to the destination device. A value of zero shall not be considered as an error. If the DC bit equals one, a value of zero specifies that no destination blocks shall be written and the only processing to be performed is that any residual source or destination data from the previous segment shall be handled as residual data as described in 6.3.7.2. If the DC bit equals zero, a value of zero specifies that no source blocks shall be read and no source data shall be processed, but any residual destination data from a previous segment shall be written if possible to the destination in whole-block transfers, and any residual data shall be handled as described in 6.3.7.2.

The SOURCE BLOCK DEVICE LOGICAL BLOCK ADDRESS field specifies the logical block address from which the reading of data shall start.

The DESTINATION BLOCK DEVICE LOGICAL BLOCK ADDRESS field specifies the logical block address to which the writing of data shall begin.

### 6.3.7.6 Stream device to stream device operations

The segment descriptor format shown in table 116 is used by the copy operations that move data from a stream device to a stream device.

**Table 116 — Stream device to stream device segment descriptor**

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (03h or 0Eh)							
1	Reserved							CAT
2	(MSB)	DESCRIPTOR LENGTH (0010h)						
3								
4	(MSB)	SOURCE TARGET DESCRIPTOR INDEX						
5								
6	(MSB)	DESTINATION TARGET DESCRIPTOR INDEX						
7								
8	Reserved							
9	(MSB)	SOURCE STREAM DEVICE TRANSFER LENGTH						
10								
12	Reserved							
13	(MSB)	DESTINATION STREAM DEVICE TRANSFER LENGTH						
14								
16	(MSB)	BYTE COUNT						
19								

The DESCRIPTOR TYPE CODE field is described in 6.3.5 and 6.3.7.1. Two DESCRIPTOR TYPE CODE values use the segment descriptor format shown in table 116 and described in this subclause.

For descriptor type code 03h (stream→ stream) or descriptor type code 0Eh (stream→ stream+application client), the copy manager shall copy the data from the source stream device identified by the SOURCE TARGET DESCRIPTOR INDEX field to the destination stream device identified by the DESTINATION TARGET DESCRIPTOR INDEX field. Data shall be read from the source stream device starting at the current position of the source stream device. Data shall be written to the destination stream device starting at the current position of the destination stream device. The BYTE COUNT field defines the number of bytes to be processed (see 6.3.7.2) by the copy manager. The copy manager shall perform read operations as necessary to supply the source data, and as many write operations as possible using the destination data.

For descriptor type code 0Eh (stream→ stream+application client), the copy manager also shall hold a copy of the processed data for delivery to the application client upon completion of the EXTENDED COPY command in response to a RECEIVE COPY RESULTS command with RECEIVE DATA service action as described in 6.18.3. The minimum amount of held data supported by the copy manager is returned in the response data for the RECEIVE COPY RESULTS command with OPERATING PARAMETERS service action (see 6.18.4). If the copy



manager supports the 0Eh descriptor type code, it also shall support the RECEIVE COPY RESULTS command with RECEIVE DATA service action.

The CAT bit is described in 6.3.7.2.

The DESCRIPTOR LENGTH field shall contain 16 (0010h). The SOURCE TARGET DESCRIPTOR INDEX and DESTINATION TARGET DESCRIPTOR INDEX fields are described in 6.3.7.1.

The SOURCE STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be read from the source stream device on each read operation. See 6.3.6.5 for a description of how data in the SOURCE STREAM DEVICE TRANSFER LENGTH field in the segment descriptor interacts with data in the STREAM BLOCK LENGTH field in the device type specific target descriptor parameters for the source sequential-access device type.

The DESTINATION STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be written to the destination stream device on each write operation. See 6.3.6.5 for a description of how data in the DESTINATION STREAM DEVICE TRANSFER LENGTH field in the segment descriptor interacts with data in the STREAM BLOCK LENGTH field in the device type specific target descriptor parameters for the destination sequential-access device type.

The BYTE COUNT field specifies the number of bytes that shall be processed for this segment descriptor. A value of zero shall not be considered as an error, and specifies that no source blocks shall be read and no source data shall be processed. However, a value of zero specifies that any residual destination data from a previous segment shall be written if possible to the destination in whole-block transfers, and any residual data shall be handled as described in 6.3.7.2.

### 6.3.7.7 Inline data to stream device operation

The segment descriptor format shown in table 117 instructs the copy manager to write inline data from the EXTENDED COPY parameter list to a stream device.

**Table 117 — Inline data to stream device segment descriptor**

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (04h)							
1	Reserved							CAT
2	(MSB)	DESCRIPTOR LENGTH (0010h)						
3								(LSB)
4	Reserved							
5	Reserved							
6	(MSB)	DESTINATION TARGET DESCRIPTOR INDEX						
7								(LSB)
8	Reserved							
9	(MSB)							
10	STREAM DEVICE TRANSFER LENGTH							
11								(LSB)
12	(MSB)	INLINE DATA OFFSET						
15								(LSB)
16	(MSB)	INLINE DATA NUMBER OF BYTES						
19								(LSB)

The DESCRIPTOR TYPE CODE field is described in 6.3.5 and 6.3.7.1. Descriptor type code 04h (inline→ stream) instructs the copy manager to write inline data from the EXTENDED COPY parameter list to a stream device. The inline data shall be read from the optional inline data at the end of the EXTENDED COPY parameter list. The data shall be written to the destination stream device identified by the DESTINATION TARGET DESCRIPTOR INDEX field starting at the current position of the stream device. Any residual destination data from a previous segment descriptor shall be written before the data of the current segment descriptor. Any residual source data from a previous segment descriptor shall not be processed (see 6.3.7.2), and shall be handled as residual source data.

The CAT bit is described in 6.3.7.2.

The DESCRIPTOR LENGTH field shall contain 16 (0010h). The DESTINATION TARGET DESCRIPTOR INDEX field is described in 6.3.7.1.

The STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be written to the stream device on each write operation. See 6.3.6.5 for a description of how data in the STREAM DEVICE TRANSFER LENGTH field in the segment descriptor interacts with data in the STREAM BLOCK LENGTH field in the device type specific target descriptor parameters for the destination sequential-access device type.

The value in the INLINE DATA OFFSET field is added to the location of the first byte of inline data in the EXTENDED COPY parameter list (see table 100) to locate the first byte of inline data to be written to the stream device. The INLINE DATA OFFSET value shall be a multiple of 4.

The INLINE DATA NUMBER OF BYTES field specifies the number of bytes of inline data that are to be transferred to the stream device. A value of zero shall not be considered an error.

If the sum of the INLINE DATA OFFSET and the INLINE DATA NUMBER OF BYTES values exceeds the value in the INLINE DATA LENGTH field (see table 100), the copy manager shall terminate the command with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INLINE DATA LENGTH EXCEEDED.

### 6.3.7.8 Embedded data to stream device operation

The segment descriptor format shown in table 118 instructs the copy manager to write embedded data from the segment descriptor to a stream device.

**Table 118 — Embedded data to stream device segment descriptor**

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (05h)							
1	Reserved							CAT
2	(MSB)	DESCRIPTOR LENGTH (n-3)						
3								(LSB)
4	Reserved							
5	Reserved							
6	(MSB)	DESTINATION TARGET DESCRIPTOR INDEX						
7								(LSB)
8	Reserved							
9	(MSB)							
10	STREAM DEVICE TRANSFER LENGTH							
11								(LSB)
12	(MSB)	EMBEDDED DATA NUMBER OF BYTES						
13								(LSB)
14	Reserved							
15								
16								
n	EMBEDDED DATA							

The DESCRIPTOR TYPE CODE field is described in 6.3.5 and 6.3.7.1. Descriptor type code 05h (embedded→ stream) instructs the copy manager to write embedded data from the segment descriptor to a stream device. The embedded data shall be read from the segment descriptor. The data shall be written to the destination stream device identified by the DESTINATION TARGET DESCRIPTOR INDEX field starting at the current position of the stream device. Any residual destination data from a previous segment descriptor shall be written before the data of the current segment descriptor. Any residual source data from a previous segment descriptor shall not be processed (see 6.3.7.2), and shall be handled as residual source data.

The CAT bit is described in 6.3.7.2.

The DESCRIPTOR LENGTH field shall contain the length in bytes of the fields that follow the DESCRIPTOR LENGTH field, including the embedded data. The value in the DESCRIPTOR LENGTH field shall be a multiple of 4.

The DESTINATION TARGET DESCRIPTOR INDEX field is described in 6.3.7.1.

The STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be written to the stream device on each write operation. See 6.3.6.5 for a description of how data in the STREAM DEVICE TRANSFER LENGTH field in the segment descriptor interacts with data in the STREAM BLOCK LENGTH field in the device type specific target descriptor parameters for the destination sequential-access device type.

The EMBEDDED DATA NUMBER OF BYTES field specifies the number of bytes of embedded data that are to be transferred to the stream device. A value of zero shall not be considered an error. The EMBEDDED DATA NUMBER OF BYTES value shall be less than or equal to the DESCRIPTOR LENGTH value minus 12.

### 6.3.7.9 Stream device to discard operation

The segment descriptor format shown in table 119 instructs the copy manager to read data from a stream device and not copy it to any destination device.

**Table 119 — Stream device to discard segment descriptor**

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (06h or 0Fh)							
1	Reserved							CAT
2	(MSB)	DESCRIPTOR LENGTH (000Ch)						
3								(LSB)
4	(MSB)	SOURCE TARGET DESCRIPTOR INDEX						
5								(LSB)
6	Reserved							
7	Reserved							
8	Reserved							
9	(MSB)							
10	STREAM DEVICE TRANSFER LENGTH							
11								(LSB)
12	(MSB)	NUMBER OF BYTES						
15								(LSB)

The DESCRIPTOR TYPE CODE field is described in 6.3.5 and 6.3.7.1. Two DESCRIPTOR TYPE CODE values use the segment descriptor format shown in table 119 and described in this subclause.

For descriptor type code 06h (stream→ discard) or descriptor type code 0Fh (stream→ discard+application client), the copy manager shall read data as necessary from the source stream device identified by the SOURCE TARGET DESCRIPTOR INDEX field starting at the current position of the source stream device. The number of bytes specified by the NUMBER OF BYTES field shall be removed from the source data, starting with any residual source data from the previous segment.

For descriptor type code 06h (stream→ discard) the removed data shall be discarded and not written to any destination device. For descriptor type code 0Fh (stream→ discard+application client) the removed data shall be held for delivery to the application client upon completion of the EXTENDED COPY command in response to a RECEIVE COPY RESULTS command with RECEIVE DATA service action as described in 6.18.3. The minimum amount of held data supported by the copy manager is returned in the response data for the RECEIVE COPY RESULTS command with OPERATING PARAMETERS service action (see 6.18.4). If the copy manager supports the 0Fh (stream→ discard+application client) descriptor type code, it also shall support the RECEIVE COPY RESULTS command with RECEIVE DATA service action.

The CAT bit is described in 6.3.7.2.

The DESCRIPTOR LENGTH field shall contain 12 (000Ch). The DESTINATION TARGET DESCRIPTOR INDEX field is described in 6.3.7.1.

The SOURCE STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be read from the source stream device on each read operation. See 6.3.6.5 for a description of how data in the SOURCE STREAM DEVICE TRANSFER LENGTH field in the segment descriptor interacts with data in the STREAM BLOCK LENGTH field in the device type specific target descriptor parameters for the source sequential-access device type.

The NUMBER OF BYTES field specifies the number of bytes to be removed from the source data.

#### 6.3.7.10 Verify device operation

The segment descriptor format shown in table 120 instructs the copy manager to verify the accessibility of a SCSI device.

**Table 120 — Verify device operation segment descriptor**

Bit Byte	7	6	5	4	3	2	1	0						
0	DESCRIPTOR TYPE CODE (07h)													
1	Reserved													
2	(MSB)	DESCRIPTOR LENGTH (0008h)						(LSB)						
3														
4	(MSB)	SOURCE TARGET DESCRIPTOR INDEX						(LSB)						
5														
6														
7								Reserved						
8								Reserved		TUR				
9								Reserved						
11														

The DESCRIPTOR TYPE CODE field is described in 6.3.5 and 6.3.7.1. Descriptor type code 07h instructs the copy manager to verify the accessibility of the device identified by the SOURCE TARGET DESCRIPTOR INDEX field.

The DESCRIPTOR LENGTH field shall contain 8 (0008h). The SOURCE TARGET DESCRIPTOR INDEX field is described in 6.3.7.1.

Support for a value of one in the TUR (Test Unit Ready) bit is optional. If setting the TUR bit to one is supported and the TUR bit is set to one, then a TEST UNIT READY command (see 6.37) shall be used to determine the readiness of the device. If setting the TUR to one is not supported and the TUR bit is set to one, then the EXTENDED COPY command shall be terminated with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INVALID FIELD IN PARAMETER LIST. The SENSE-KEY SPECIFIC field shall be set as described in 6.3.3. If the TUR bit is set to zero, then the accessibility should be verified without disturbing established unit attention conditions or ACA conditions (e.g., using the INQUIRY command (see 6.4)).

#### 6.3.7.11 Block device with offset to stream device operation

The segment descriptor format shown in table 121 is used to instruct the copy manager to move data from a block device with a byte offset to a stream device or vice versa.

**Table 121 — Block device with offset to or from stream device segment descriptor**

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (08h or 09h)							
1	Reserved							CAT
2	(MSB)	DESCRIPTOR LENGTH (0018h)						(LSB)
3								
4	(MSB)	SOURCE TARGET DESCRIPTOR INDEX						(LSB)
5								
6	(MSB)	DESTINATION TARGET DESCRIPTOR INDEX						(LSB)
7								
8	Reserved							
9	(MSB)	STREAM DEVICE TRANSFER LENGTH						(LSB)
10								
11		NUMBER OF BYTES						(LSB)
12	(MSB)							
15		BLOCK DEVICE LOGICAL BLOCK ADDRESS						(LSB)
16	(MSB)							
23		Reserved						(LSB)
24								
25	Reserved							
26	(MSB)	BLOCK DEVICE BYTE OFFSET						(LSB)
27								

The DESCRIPTOR TYPE CODE field is described in 6.3.5 and 6.3.7.1. Descriptor type code 08h (block<o>→ stream) instructs the copy manager to copy the data from the source block device identified by the SOURCE TARGET DESCRIPTOR INDEX field to the destination stream device identified by the DESTINATION TARGET DESCRIPTOR INDEX field using data starting at the location identified by the BLOCK DEVICE BYTE OFFSET field in the logical block identified by the BLOCK DEVICE LOGICAL BLOCK ADDRESS field and continuing for the number of bytes specified in the NUMBER OF BYTES field. The data shall be written to the stream device starting at the current position of the media.

The CAT bit is described in 6.3.7.2.

The DESCRIPTOR LENGTH field shall contain 24 (0018h). The SOURCE TARGET DESCRIPTOR INDEX and DESTINATION TARGET DESCRIPTOR INDEX fields are described in 6.3.7.1.

The STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be written on each write operation to the stream device. See 6.3.6.5 for a description of how data in the STREAM DEVICE TRANSFER LENGTH field in the segment descriptor interacts with data in the STREAM BLOCK LENGTH field in the device type specific target descriptor parameters for the sequential-access device type.

The NUMBER OF BYTES field specifies the number bytes to be read. A value of zero specifies that no bytes shall be transferred in this segment. This shall not be considered as an error.

The BLOCK DEVICE LOGICAL BLOCK ADDRESS field specifies the starting logical block address on the source block device for this segment.

The BLOCK DEVICE BYTE OFFSET field specifies the offset into the first source block at which to begin reading bytes.

#### **6.3.7.12 Stream device to block device with offset operation**

The segment descriptor format shown in table 121 (see 6.3.7.11) also is used to instruct the copy manager to move data from a stream device to a block device with a byte offset.

The DESCRIPTOR TYPE CODE field is described in 6.3.5 and 6.3.7.1. Descriptor type code 09h (stream→ block<0>) instructs the copy manager to copy the data from the source stream device identified by the SOURCE TARGET DESCRIPTOR INDEX field to the destination block device identified by the DESTINATION TARGET DESCRIPTOR INDEX field using the stream data starting at the current position of the stream device. The data shall be written starting at the location identified by the BLOCK DEVICE BYTE OFFSET field in the logical block identified by the BLOCK DEVICE LOGICAL BLOCK ADDRESS field and continuing for the number of bytes specified in the NUMBER OF BYTES field.

The content of the starting logical block on the destination device before the starting offset shall be preserved. The content on the ending logical block beyond the end of the transfer shall be preserved. The copy manager may implement this operation by reading the starting and ending logical blocks, modifying a portion of the blocks as required, and writing the full blocks to the destination device.

The CAT bit is described in 6.3.7.2.

The DESCRIPTOR LENGTH field shall contain 24 (0018h). The SOURCE TARGET DESCRIPTOR INDEX and DESTINATION TARGET DESCRIPTOR INDEX fields are described in 6.3.7.1.

The STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be written on each write operation to the stream device. See 6.3.6.5 for a description of how data in the STREAM DEVICE TRANSFER LENGTH field in the segment descriptor interacts with data in the STREAM BLOCK LENGTH field in the device type specific target descriptor parameters for the sequential-access device type.

The NUMBER OF BYTES field specifies the number bytes to be read. A value of zero specifies that no bytes shall be transferred in this segment. This shall not be considered as an error.

The BLOCK DEVICE LOGICAL BLOCK ADDRESS field specifies the starting logical block address on the destination block device for this segment.

The BLOCK DEVICE BYTE OFFSET field specifies the offset into the first destination block at which to begin writing data to the destination block device.

### 6.3.7.13 Block device with offset to block device with offset operation

The segment descriptor format shown in table 122 instructs the copy manager to move data from a block device with a byte offset to a block device with a byte offset.

**Table 122 — Block device with offset to block device with offset segment descriptor**

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (0Ah)							
1	Reserved							CAT
2	(MSB)	DESCRIPTOR LENGTH (001Ch)						(LSB)
3								
4	(MSB)	SOURCE TARGET DESCRIPTOR INDEX						(LSB)
5								
6	(MSB)	DESTINATION TARGET DESCRIPTOR INDEX						(LSB)
7								
8	(MSB)	NUMBER OF BYTES						(LSB)
11								
12	(MSB)	SOURCE BLOCK DEVICE LOGICAL BLOCK ADDRESS						(LSB)
19								
20	(MSB)	DESTINATION BLOCK DEVICE LOGICAL BLOCK ADDRESS						(LSB)
27								
28	(MSB)	SOURCE BLOCK DEVICE BYTE OFFSET						(LSB)
29								
30	(MSB)	DESTINATION BLOCK DEVICE BYTE OFFSET						(LSB)
31								

The DESCRIPTOR TYPE CODE field is described in 6.3.5 and 6.3.7.1. Descriptor type code 0Ah (block<o>→block<o>) instructs the copy manager to copy the data from the source block device identified by the SOURCE TARGET DESCRIPTOR INDEX field to the destination block device identified by the DESTINATION TARGET DESCRIPTOR INDEX field using data starting at the location identified by the source BLOCK DEVICE BYTE OFFSET field in the logical block identified by the SOURCE BLOCK DEVICE LOGICAL BLOCK ADDRESS field and continuing for the number of bytes specified in the NUMBER OF BYTES field. The data shall be written starting at the location identified by the DESTINATION BLOCK DEVICE BYTE OFFSET field in the logical block identified by the DESTINATION BLOCK DEVICE LOGICAL BLOCK ADDRESS field.

The content of the starting logical block on the destination device before the starting offset shall be preserved. The content on the ending logical block beyond the end of the transfer shall be preserved. The copy manager may implement this operation by reading the starting and ending logical blocks, modifying a portion of the blocks as required, and writing the full blocks to the destination device.

The CAT bit is described in 6.3.7.2.

The DESCRIPTOR LENGTH field shall contain 28 (001Ch). The SOURCE TARGET DESCRIPTOR INDEX and DESTINATION TARGET DESCRIPTOR INDEX fields are described in 6.3.7.1.



The NUMBER OF BYTES field specifies the number bytes to be read. A value of zero specifies that no bytes shall be transferred in this segment. This shall not be considered as an error.

The SOURCE BLOCK DEVICE LOGICAL BLOCK ADDRESS field specifies the starting address on the source block device for this segment.

The DESTINATION BLOCK DEVICE LOGICAL BLOCK ADDRESS field specifies the starting logical block address on the destination block device for this segment.

The SOURCE BLOCK DEVICE BYTE OFFSET field specifies the offset into the first source block at which to begin reading bytes.

The DESTINATION BLOCK DEVICE BYTE OFFSET field specifies the offset into the first destination block at which to begin writing data to the destination block device.

#### 6.3.7.14 Write filemarks operation

The segment descriptor format shown in table 123 instructs the copy manager to write filemarks or setmarks on the destination tape device.

**Table 123 — Write filemarks operation segment descriptor**

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (10h)							
1	Reserved							
2	(MSB)	DESCRIPTOR LENGTH (0008h)						(LSB)
3								
4	Reserved							
5	Reserved							
6	(MSB)	DESTINATION TARGET DESCRIPTOR INDEX						(LSB)
7								
8	Reserved						WSMK	IMMED
9	(MSB)	TRANSFER LENGTH						(LSB)
10								
11								

The DESCRIPTOR TYPE CODE field is described in 6.3.5 and 6.3.7.1. Descriptor type code 10h (filemark→ tape) instructs the copy manager to write filemarks or setmarks to the destination tape device identified by the DESTINATION TARGET DESCRIPTOR INDEX field starting at the current position of the tape device. If the PERIPHERAL DEVICE TYPE field in the target descriptor identified by the DESTINATION TARGET DESCRIPTOR INDEX field does not contain 01h, the copy manager shall terminate the command with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INVALID OPERATION FOR COPY SOURCE OR DESTINATION.

The DESCRIPTOR LENGTH field shall contain 8 (0008h). The DESTINATION TARGET DESCRIPTOR INDEX field is described in 6.3.7.1.

If the write setmark (WSMK) bit is set to one, the TRANSFER LENGTH field specifies the number of setmarks to be written. If the WSMK bit is set to zero, the TRANSFER LENGTH field specifies the number of filemarks to be written.

If the immediate (IMMED) bit in the segment descriptor is set to one, then the copy manager shall issue a WRITE FILEMARKS command to the destination tape device with the immediate bit is set to one. If the IMMED bit is set to zero, then the copy manager shall issue a WRITE FILEMARKS command to the destination tape device with the immediate bit is set to zero.

### 6.3.7.15 Space operation

The segment descriptor format shown in table 124 instructs the copy manager to send a SPACE command (see SSC-2) to the destination tape device.

**Table 124 — Space operation segment descriptor**

Bit Byte	7	6	5	4	3	2	1	0					
0	DESCRIPTOR TYPE CODE (11h)												
1	Reserved												
2	(MSB)	DESCRIPTOR LENGTH (0008h)						(LSB)					
3													
4	Reserved												
5	Reserved												
6	(MSB)	DESTINATION TARGET DESCRIPTOR INDEX						(LSB)					
7													
8	Reserved				CODE								
9	(MSB)	COUNT						(LSB)					
10													
11													

The DESCRIPTOR TYPE CODE field is described in 6.3.5 and 6.3.7.1. Descriptor type code 11h (space→ tape) instructs the copy manager to send a SPACE command to the destination tape device identified by the DESTINATION TARGET DESCRIPTOR INDEX field. If the PERIPHERAL DEVICE TYPE field in the target descriptor identified by the DESTINATION TARGET DESCRIPTOR INDEX field does not contain 01h, the copy manager shall terminate the command with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INVALID OPERATION FOR COPY SOURCE OR DESTINATION.

The DESCRIPTOR LENGTH field shall contain 8 (0008h). The DESTINATION TARGET DESCRIPTOR INDEX field is described in 6.3.7.1.

The CODE and COUNT fields contents in the SPACE command sent to the destination tape device shall be copied from the CODE and COUNT fields in the segment descriptor. All other fields in the SPACE command sent to the destination tape device that affect the positioning of the tape shall be set to zero.

### 6.3.7.16 Locate operation

The segment descriptor format shown in table 125 instructs the copy manager to send a LOCATE command (see SSC-2) to the destination tape device.

**Table 125 — Locate operation segment descriptor**

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (12h)							
1	Reserved							
2	(MSB)	DESCRIPTOR LENGTH (0008h)						(LSB)
3								
4	Reserved							
5	Reserved							
6	(MSB)	DESTINATION TARGET DESCRIPTOR INDEX						(LSB)
7								
8	(MSB)	BLOCK ADDRESS						(LSB)
11								

The DESCRIPTOR TYPE CODE field is described in 6.3.5 and 6.3.7.1. Descriptor type code 12h (locate→ tape) instructs the copy manager to send a LOCATE command to the destination tape device identified by the DESTINATION TARGET DESCRIPTOR INDEX field. If the PERIPHERAL DEVICE TYPE field in the target descriptor identified by the DESTINATION TARGET DESCRIPTOR INDEX field does not contain 01h, the copy manager shall terminate the command with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INVALID OPERATION FOR COPY SOURCE OR DESTINATION.

The DESCRIPTOR LENGTH field shall contain 8 (0008h). The DESTINATION TARGET DESCRIPTOR INDEX field is described in 6.3.7.1.

The BLOCK ADDRESS field contents in the LOCATE command sent to the destination tape device shall be copied from the BLOCK ADDRESS field in the segment descriptor. All other fields in the LOCATE command sent to the destination tape device that affect the positioning of the tape shall be set to zero.

NOTE 26 - The restrictions described above for the LOCATE command limit the operation to locating SCSI logical block addresses in the current tape partition.

### 6.3.7.17 Tape device image copy operation

The segment descriptor format shown in table 126 instructs the copy manager to perform an image copy from the source tape device to the destination tape device.

**Table 126 — Tape device image copy segment descriptor**

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (13h)							
1	Reserved							
2	(MSB)	DESCRIPTOR LENGTH (0008h)						
3								(LSB)
4	(MSB)	SOURCE TARGET DESCRIPTOR INDEX						
5								(LSB)
6	(MSB)	DESTINATION TARGET DESCRIPTOR INDEX						
7								(LSB)
8	(MSB)	COUNT						
11								(LSB)

The DESCRIPTOR TYPE CODE field is described in 6.3.5 and 6.3.7.1. Descriptor type code 13h (<i>tape→ <i>tape) instructs the copy manager to create a compatible image of the source device medium identified by the SOURCE TARGET DESCRIPTOR INDEX field on the destination device medium identified by the DESTINATION TARGET DESCRIPTOR INDEX field beginning at their current positions. If the PERIPHERAL DEVICE TYPE field in the target descriptor identified by the SOURCE TARGET DESCRIPTOR INDEX field or the DESTINATION TARGET DESCRIPTOR INDEX field does not contain 01h, the copy manager shall terminate the command with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INVALID OPERATION FOR COPY SOURCE OR DESTINATION.

The DESCRIPTOR LENGTH field shall contain 8 (0008h). The SOURCE TARGET DESCRIPTOR INDEX and DESTINATION TARGET DESCRIPTOR INDEX fields are described in 6.3.7.1.

The tape image copy operation terminates when:

- The source device encounters an end-of-partition as defined by the source device;
- The source device encounters an end-of-data as defined by the source device (i.e., BLANK CHECK sense key);
- The copy manager has copied the number of consecutive filemarks specified in the COUNT field from the source device to the destination device; or
- The copy manager has copied the number of consecutive filemarks and/or setmarks specified in the COUNT field from the source device to the destination device, if the RSMK bit in the Device Configuration mode page (see SSC-2) of the source device is set to one.

A COUNT field of zero specifies that the EXTENDED COPY command shall not terminate due to any number of consecutive filemarks or setmarks. Other error or exception conditions (e.g., early-warning, end-of-partition on destination device) may cause the EXTENDED COPY command to terminate prior to completion. If this occurs, the residue shall not be calculated and the INFORMATION field in the sense data shall be set to zero.

### 6.3.7.18 Register persistent reservation key operation

The segment descriptor format shown in table 127 instructs the copy manager to register an I\_T nexus using the reservation key (see 5.7.7) specified by the RESERVATION KEY field with the SCSI target device specified by the DESTINATION TARGET DESCRIPTOR INDEX field.

**Table 127 — Register persistent reservation key segment descriptor**

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (14h)							
1	Reserved							
2	(MSB)	DESCRIPTOR LENGTH (0018h)						(LSB)
3								
4	Reserved							
5	Reserved							
6	(MSB)	DESTINATION TARGET DESCRIPTOR INDEX						(LSB)
7								
8	(MSB)	RESERVATION KEY						(LSB)
15								
16	(MSB)	SERVICE ACTION RESERVATION KEY						(LSB)
23								
24	Reserved							
27								

The DESCRIPTOR TYPE CODE field is described in 6.3.5 and 6.3.7.1. Descriptor type code 14h instructs the copy manager to register an I\_T nexus using the reservation key specified by the RESERVATION KEY field with the SCSI target device identified by the DESTINATION TARGET DESCRIPTOR INDEX field using a PERSISTENT RESERVE OUT command with a REGISTER service action (see 6.14.2).

The DESCRIPTOR LENGTH field shall contain 24 (0018h). The DESTINATION TARGET DESCRIPTOR INDEX field is described in 6.3.7.1.

The RESERVATION KEY and SERVICE ACTION RESERVATION KEY field contents in the PERSISTENT RESERVE OUT command sent to the destination device shall be copied from the RESERVATION KEY and SERVICE ACTION RESERVATION KEY fields in the segment descriptor.

The application client sending the EXTENDED COPY command may need to remove the reservation key held by the copy manager as described in 5.7.11 prior to sending the EXTENDED COPY command.

### 6.3.7.19 Third party persistent reservations source I\_T nexus

The segment descriptor format shown in table 128 instructs the copy manager to send a PERSISTENT RESERVATION OUT command with REGISTER AND MOVE service action (see 5.7.8) with the specified I\_T nexus after all other segment descriptors have been processed. If an error is detected any time after receiving a third party persistent source reservation I\_T nexus segment descriptor, the PERSISTENT RESERVATION OUT command REGISTER AND MOVE service action shall be processed before status is returned for the EXTENDED COPY command.

This segment descriptor should be placed at or near the beginning of the list of segment descriptors to assure the copy manager processes the PERSISTENT RESERVATION OUT command with REGISTER AND MOVE service action in the event of an error that terminates the processing of segment descriptors. If an error is detected in a segment descriptor and third party persistent reservations source I\_T nexus segment descriptor has not been processed, the copy manager shall not send a PERSISTENT RESERVATION OUT command with REGISTER AND MOVE service action.

Placing more than one source third party persistent reservations source I\_T nexus segment descriptor in the list of descriptors is not an error. All source third party persistent reservations source I\_T nexus segment descriptors known to the copy manager shall be processed after all other segment descriptors have been processed.

**Table 128 — Third party persistent reservations source I\_T nexus segment descriptor**

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (15h)							
1	Reserved							
2	(MSB)	DESCRIPTOR LENGTH (n-3)						(LSB)
3								
4	Reserved							
5	Reserved							
6	(MSB)	DESTINATION TARGET DESCRIPTOR INDEX						(LSB)
7								
8	(MSB)	RESERVATION KEY						(LSB)
15								
16	(MSB)	SERVICE ACTION RESERVATION KEY						(LSB)
23								
24	Reserved							
25	Reserved					UNREG		APTPL
26	(MSB)	RELATIVE TARGET PORT IDENTIFIER						(LSB)
27								
28	(MSB)	TRANSPORTID PARAMETER DATA LENGTH (n-31)						(LSB)
31								
32	TransportID							
n								

The DESCRIPTOR TYPE CODE field is described in 6.3.5 and 6.3.7.1. Descriptor type code 15h instructs the copy manager to send PERSISTENT RESERVATION OUT command with REGISTER AND MOVE service action (see 6.14) to the target port identified by the DESTINATION TARGET DESCRIPTOR INDEX field.

The DESCRIPTOR LENGTH field shall contain the length in bytes of the fields that follow the DESCRIPTOR LENGTH field. The value in the DESCRIPTOR LENGTH field shall be a multiple of 4.

If the PERIPHERAL DEVICE TYPE field in the target descriptor identified by the DESTINATION TARGET DESCRIPTOR INDEX field does not contain 01h, the copy manager shall terminate the command with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INVALID OPERATION FOR COPY SOURCE OR DESTINATION.

Bytes 8 to n of the segment descriptor shall be sent as the parameter list (see 6.14.4) for the PERSISTENT RESERVE OUT command with REGISTER AND MOVE service action.

For a description of the RESERVATION KEY field, SERVICE ACTION RESERVATION KEY field, UNREG bit, APTPL bit, RELATIVE TARGET PORT IDENTIFIER field, TRANSPORTID DESCRIPTOR LENGTH field, and TransportID, see 6.14.4.

## 6.4 INQUIRY command

### 6.4.1 INQUIRY command introduction

The INQUIRY command (see table 129) requests that information regarding the logical unit and SCSI target device be sent to the application client.

**Table 129 — INQUIRY command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (12h)							
1	Reserved						Obsolete	EVPD
2	PAGE CODE							
3	(MSB)	ALLOCATION LENGTH						(LSB)
4								
5	CONTROL							

An enable vital product data (EVPD) bit set to one specifies that the device server shall return the vital product data specified by the PAGE CODE field (see 6.4.4).

If the EVPD bit is set to zero, the device server shall return the standard INQUIRY data (see 6.4.2). If the PAGE CODE field is not set to zero when the EVPD bit is set to zero, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

When the EVPD bit is set to one, the PAGE CODE field specifies which page of vital product data information the device server shall return (see 7.7).

The ALLOCATION LENGTH field is defined in 4.3.5.6.

In response to an INQUIRY command received by an incorrect logical unit, the SCSI target device shall return the INQUIRY data with the peripheral qualifier set to the value defined in 6.4.2. The INQUIRY command shall return CHECK CONDITION status only when the device server is unable to return the requested INQUIRY data.

If an INQUIRY command is received from an initiator port with a pending unit attention condition (i.e., before the device server reports CHECK CONDITION status), the device server shall perform the INQUIRY command and shall not clear the unit attention condition (see SAM-4).

The INQUIRY data should be returned even though the device server is not ready for other commands. The standard INQUIRY data should be available without incurring any media access delays. If the device server does store some of the standard INQUIRY data or VPD data on the media, it may return ASCII spaces (20h) in ASCII fields and zeros in other fields until the data is available from the media.

The INQUIRY data may change as the SCSI target device and its logical units perform their initialization sequence. (E.g., logical units may provide a minimum command set from nonvolatile memory until they load the final firmware from the media. After the firmware has been loaded, more options may be supported and therefore different INQUIRY data may be returned.)



If the INQUIRY data changes for any reason, the device server shall establish a unit attention condition for the initiator port associated with every I\_T nexus (see SAM-4), with the additional sense code set to INQUIRY DATA HAS CHANGED.

NOTE 27 - The INQUIRY command may be used by an application client after a hard reset or power on condition to determine the device types for system configuration.

### 6.4.2 Standard INQUIRY data

The standard INQUIRY data (see table 130) shall contain at least 36 bytes.

**Table 130 — Standard INQUIRY data format**

Bit Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	RMB	Reserved						
2	VERSION							
3	Obsolete	Obsolete	NORMACA	HiSUP	RESPONSE DATA FORMAT			
4	ADDITIONAL LENGTH (n-4)							
5	SCCS	ACC	TPGS		3PC	Reserved		PROTECT
6	Obsolete	ENCSERV	VS	MULTIP	Obsolete	Obsolete	Obsolete	ADDR16 <sup>a</sup>
7	Obsolete	Obsolete	WBUS16 <sup>a</sup>	SYNC <sup>a</sup>	Obsolete	Obsolete	CMDQUE	VS
8	(MSB) T10 VENDOR IDENTIFICATION (LSB)							
15								
16	(MSB) PRODUCT IDENTIFICATION (LSB)							
31								
32	(MSB) PRODUCT REVISION LEVEL (LSB)							
35								
36	Vendor specific							
55								
56	Reserved				CLOCKING <sup>a</sup>		QAS <sup>a</sup>	IUS <sup>a</sup>
57	Reserved							
58	(MSB) VERSION DESCRIPTOR 1 (LSB)							
59								
	⋮							
72	(MSB) VERSION DESCRIPTOR 8 (LSB)							
73								
74	Reserved							
95								
	Vendor specific parameters							
96	Vendor specific							
n								
<sup>a</sup> The meanings of these fields are specific to SPI-5 (see 6.4.3). For SCSI transport protocols other than the SCSI Parallel Interface, these fields are reserved.								

The PERIPHERAL QUALIFIER field and PERIPHERAL DEVICE TYPE field identify the peripheral device connected to the logical unit. If the SCSI target device is not capable of supporting a peripheral device connected to this logical unit, the device server shall set these fields to 7Fh (i.e., PERIPHERAL QUALIFIER field set to 011b and PERIPHERAL DEVICE TYPE field set to 1Fh).

The peripheral qualifier is defined in table 131 and the peripheral device type is defined in table 132.

**Table 131 — Peripheral qualifier**

Qualifier	Description
000b	A peripheral device having the specified peripheral device type is connected to this logical unit. If the device server is unable to determine whether or not a peripheral device is connected, it also shall use this peripheral qualifier. This peripheral qualifier does not mean that the peripheral device connected to the logical unit is ready for access.
001b	A peripheral device having the specified peripheral device type is not connected to this logical unit. However, the device server is capable of supporting the specified peripheral device type on this logical unit.
010b	Reserved
011b	The device server is not capable of supporting a peripheral device on this logical unit. For this peripheral qualifier the peripheral device type shall be set to 1Fh. All other peripheral device type values are reserved for this peripheral qualifier.
100b to 111b	Vendor specific

**Table 132 — Peripheral device type (part 1 of 2)**

Code	Reference <sup>a</sup>	Description
00h	SBC-3	Direct access block device (e.g., magnetic disk)
01h	SSC-3	Sequential-access device (e.g., magnetic tape)
02h	SSC	Printer device
03h	SPC-2	Processor device
04h	SBC	Write-once device (e.g., some optical disks)
05h	MMC-5	CD/DVD device
06h		Scanner device (obsolete)
07h	SBC	Optical memory device (e.g., some optical disks)
08h	SMC-3	Media changer device (e.g., jukeboxes)
09h		Communications device (obsolete)
0Ah to 0Bh		Obsolete
0Ch	SCC-2	Storage array controller device (e.g., RAID)
0Dh	SES	Enclosure services device
0Eh	RBC	Simplified direct-access device (e.g., magnetic disk)
0Fh	OCRW	Optical card reader/writer device
10h	BCC	Bridge Controller Commands
11h	OSD	Object-based Storage Device
<sup>a</sup> All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the listed standards. <sup>b</sup> All well known logical units use the same peripheral device type code.		

**Table 132 — Peripheral device type (part 2 of 2)**

Code	Reference <sup>a</sup>	Description
12h	ADC-2 clause 9	Automation/Drive Interface
13h		Security manager device
14h to 1Dh		Reserved
1Eh		Well known logical unit <sup>b</sup>
1Fh		Unknown or no device type
<sup>a</sup> All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the listed standards.		
<sup>b</sup> All well known logical units use the same peripheral device type code.		

A removable medium (RMB) bit set to zero indicates that the medium is not removable. A RMB bit set to one indicates that the medium is removable.

The VERSION field indicates the implemented version of this standard and is defined in table 133.

**Table 133 — Version**

Code	Description		
00h	The device does not claim conformance to any standard.		
02h	Obsolete		
03h	The device complies to ANSI INCITS 301-1997 (SPC).		
04h	The device complies to ANSI INCITS 351-2001 (SPC-2).		
05h	The device complies to ANSI INCITS 408-2005 (SPC-3).		
06h	The device complies to this standard.		
Code	Description	Code	Description
01h	Obsolete (SCSI=001b)	07h	Reserved
08h to 0Ch	Obsolete (ECMA=001b)	0Dh to 3Fh	Reserved
40h to 44h	Obsolete (ISO=01b)	45h to 47h	Reserved
48h to 4Ch	Obsolete (ISO=01b & ECMA=001b)	4Dh to 7Fh	Reserved
80h to 84h	Obsolete (ISO=10b)	85h to 87h	Reserved
88h to 8Ch	Obsolete (ECMA=001b)	8Dh to FFh	Reserved

The Normal ACA Supported (NORMACA) bit set to one indicates that the device server supports a NACA bit set to one in the CDB CONTROL byte and supports the ACA task attribute (see SAM-4). A NORMACA bit set to zero indicates that the device server does not support a NACA bit set to one and does not support the ACA task attribute.

A hierarchical support (HISUP) bit set to zero indicates the SCSI target device does not use the hierarchical addressing model to assign LUNs to logical units. A HISUP bit set to one indicates the SCSI target device uses the hierarchical addressing model to assign LUNs to logical units.

A RESPONSE DATA FORMAT field value of two indicates that the data shall be in the format defined in this standard. Response data format values less than two are obsolete. Response data format values greater than two are reserved.

The ADDITIONAL LENGTH field indicates the length in bytes of the remaining standard INQUIRY data. The relationship between the ADDITIONAL LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

An SCC Supported (SCCS) bit set to one indicates that the SCSI target device contains an embedded storage array controller component. See SCC-2 for details about storage array controller devices. An SCCS bit set to zero indicates that the SCSI target device does not contain an embedded storage array controller component.

An Access Controls Coordinator (ACC) bit set to one indicates that the SCSI target device contains an access controls coordinator (see 3.1.4) that may be addressed through this logical unit. An ACC bit set to zero indicates that no access controls coordinator may be addressed through this logical unit. If the SCSI target device contains an access controls coordinator that may be addressed through any logical unit other than the ACCESS CONTROLS well known logical unit (see 8.3), then the ACC bit shall be set to one for LUN 0.

The contents of the target port group support (TPGS) field (see table 134) indicate the support for asymmetric logical unit access (see 5.9).

**Table 134 — TPGS field**

Code	Description
00b	The SCSI target device does not support asymmetric logical unit access or supports a form of asymmetric access that is vendor specific. Neither the REPORT TARGET GROUPS nor the SET TARGET GROUPS commands is supported.
01b	Only implicit asymmetric logical unit access (see 5.9.2.7) is supported. The SCSI target device is capable of changing target port asymmetric access states without a SET TARGET PORT GROUPS command. The REPORT TARGET PORT GROUPS command is supported and the SET TARGET PORT GROUPS command is not supported.
10b	Only explicit asymmetric logical unit access (see 5.9.2.8) is supported. The SCSI target device only changes target port asymmetric access states as requested with the SET TARGET PORT GROUPS command. Both the REPORT TARGET PORT GROUPS command and the SET TARGET PORT GROUPS command are supported.
11b	Both explicit and implicit asymmetric logical unit access are supported. Both the REPORT TARGET PORT GROUPS command and the SET TARGET PORT GROUPS commands are supported.

A Third-Party Copy (3PC) bit set to one indicates that the SCSI target device supports third-party copy commands such as the EXTENDED COPY command (see 6.3). A 3PC bit set to zero indicates that the SCSI target device does not support such commands.

A PROTECT bit set to zero indicates that the logical unit does not support protection information (i.e., type 0 protection). A PROTECT bit set to one indicates that the logical unit supports type 1 protection, type 2 protection, or type 3 protection (see SBC-3). The SPT field (see 7.7.4) indicates which type of protection the logical unit supports.

An Enclosure Services (ENC SERV) bit set to one indicates that the SCSI target device contains an embedded enclosure services component. See SES for details about enclosure services, including a device model for an embedded enclosure services device. An ENC SERV bit set to zero indicates that the SCSI target device does not contain an embedded enclosure services component.

A Multi Port (MULTIP) bit set to one indicates that this is a multi-port (two or more ports) SCSI target device and conforms to the SCSI multi-port device requirements found in the applicable standards (e.g., SAM-4, a SCSI transport protocol standard and possibly provisions of a command standard). A MULTIP bit set to zero indicates that this SCSI target device has a single port and does not implement the multi-port requirements.

The CMDQUE bit shall be set to one indicating that the logical unit supports the task management model (see SAM-4).

The T10 VENDOR IDENTIFICATION field contains eight bytes of left-aligned ASCII data (see 4.4.1) identifying the vendor of the product. The T10 vendor identification shall be one assigned by INCITS. A list of assigned T10 vendor identifications is in Annex E and on the T10 web site (<http://www.t10.org>).

NOTE 28 - The T10 web site (<http://www.t10.org>) provides a convenient means to request an identification code.

The PRODUCT IDENTIFICATION field contains sixteen bytes of left-aligned ASCII data (see 4.4.1) defined by the vendor.

The PRODUCT REVISION LEVEL field contains four bytes of left-aligned ASCII data defined by the vendor.

The VERSION DESCRIPTOR fields provide for identifying up to eight standards to which the SCSI target device claims conformance. The value in each VERSION DESCRIPTOR field shall be selected from table 135. All version descriptor values not listed in table 135 are reserved. Technical Committee T10 of INCITS maintains an electronic copy of the information in table 135 on its world wide web site (<http://www.t10.org/>). In the event that the T10 world wide web site is no longer active, access may be possible via the INCITS world wide web site (<http://www.incits.org>), the ANSI world wide web site (<http://www.ansi.org>), the IEC site (<http://www.iec.ch/>), the ISO site (<http://www.iso.ch/>), or the ISO/IEC JTC 1 web site (<http://www.jtc1.org/>). It is recommended that the first version descriptor be used for the SCSI architecture standard, followed by the physical transport standard if any, followed by the SCSI transport protocol standard, followed by the appropriate SPC version, followed by the device type command set, followed by a secondary command set if any.

**Table 135 — Version descriptor values (part 1 of 9)**

Standard	Version Descriptor Value
ADC (no version claimed)	03C0h
ADC ANSI INCITS 403-2005	03D7h
ADC T10/1558-D revision 7	03D6h
ADC T10/1558-D revision 6	03D5h
ADC-3 (no version claimed)	0500h
ADC-2 (no version claimed)	04A0h
ADC-2 ANSI INCITS 441-2008	04ACh
ADC-2 T10/1741-D revision 7	04A7h
ADC-2 T10/1741-D revision 8	04AAh
ADP (no version claimed)	09C0h
ADT (no version claimed)	09E0h
ADT ANSI INCITS 406-2005	09FDh
ADT T10/1557-D revision 14	09FAh
ADT T10/1557-D revision 11	09F9h
ADT-2 (no version claimed)	0A20h
ATA/ATAPI-6 (no version claimed)	15E0h
ATA/ATAPI-6 ANSI INCITS 361-2002	15FDh
ATA/ATAPI-7 (no version claimed)	1600h
Annex D contains the version descriptor value assignments in numeric order.	

**Table 135 — Version descriptor values** (part 2 of 9)

<b>Standard</b>	<b>Version Descriptor Value</b>
ATA/ATAPI-7 ANSI INCITS 397-2005	161Ch
ATA/ATAPI-7 T13/1532-D revision 3	1602h
ATA/ATAPI-8 ATA8-AAM Architecture Model (no version claimed)	1620h
ATA/ATAPI-8 ATA8-APT Parallel Transport (no version claimed)	1621h
ATA/ATAPI-8 ATA8-AST Serial Transport (no version claimed)	1622h
ATA/ATAPI-8 ATA8-ACS ATA/ATAPI Command Set (no version claimed)	1623h
BCC (no version claimed)	0380h
EPI (no version claimed)	0B20h
EPI ANSI INCITS TR-23 1999	0B3Ch
EPI T10/1134 revision 16	0B3Bh
Fast-20 (no version claimed)	0AC0h
Fast-20 ANSI INCITS 277-1996	0ADCh
Fast-20 T10/1071 revision 06	0ADBh
FC 10GFC (no version claimed)	0EA0h
FC 10GFC ISO/IEC 14165-116	0EA3h
FC 10GFC ANSI INCITS 364-2003	0EA2h
FC 10GFC ANSI INCITS 364-2003 with AM1 ANSI INCITS 364/AM1-2007	0EA6h
FC-AL (no version claimed)	0D40h
FC-AL ANSI INCITS 272-1996	0D5Ch
FC-AL-2 (no version claimed)	0D60h
FC-AL-2 ANSI INCITS 332-1999 with AM1-2003 & AM2-2006	0D63h
FC-AL-2 ANSI INCITS 332-1999 with Amnd 2 AM2-2006	0D64h
FC-AL-2 ANSI INCITS 332-1999 with Amnd 1 AM1-2003	0D7Dh
FC-AL-2 ANSI INCITS 332-1999	0D7Ch
FC-AL-2 T11/1133-D revision 7.0	0D61h
FC-DA (no version claimed)	12E0h
FC-DA ANSI INCITS TR-36 2004	12E8h
FC-DA T11/1513-DT revision 3.1	12E2h
FC-DA-2 (no version claimed)	12C0h
FC-FLA (no version claimed)	1320h
FC-FLA ANSI INCITS TR-20 1998	133Ch
FC-FLA T11/1235 revision 7	133Bh
FC-FS (no version claimed)	0DA0h
FC-FS ANSI INCITS 373-2003	0DBCh
FC-FS T11/1331-D revision 1.2	0DB7h
FC-FS T11/1331-D revision 1.7	0DB8h
FC-FS-2 (no version claimed)	0E00h
Annex D contains the version descriptor value assignments in numeric order.	

**Table 135 — Version descriptor values** (part 3 of 9)

<b>Standard</b>	<b>Version Descriptor Value</b>
FC-FS-2 ANSI INCITS 242-2007 with AM1 ANSI INCITS 242/AM1-2007	0E03h
FC-FS-2 ANSI INCITS 242-2007	0E02h
FC-FS-3 (no version claimed)	0EE0h
FC-LS (no version claimed)	0E20h
FC-LS ANSI INCITS 433-2007	0E29h
FC-LS T11/1620-D revision 1.62	0E21h
FC-LS-2 (no version claimed)	0F00h
FCP (no version claimed)	08C0h
FCP ANSI INCITS 269-1996	08DCh
FCP T10/0993-D revision 12	08DBh
FC-PH (no version claimed)	0D20h
FC-PH ANSI INCITS 230-1994	0D3Bh
FC-PH ANSI INCITS 230-1994 with Amnd 1 ANSI INCITS 230/AM1-1996	0D3Ch
FC-PH-3 (no version claimed)	0D80h
FC-PH-3 ANSI INCITS 303-1998	0D9Ch
FC-PI (no version claimed)	0DC0h
FC-PI ANSI INCITS 352-2002	0DDCh
FC-PI-2 (no version claimed)	0DE0h
FC-PI-2 ANSI INCITS 404-2006	0DE4h
FC-PI-2 T11/1506-D revision 5.0	0DE2h
FC-PI-3 (no version claimed)	0E60h
FC-PI-4 (no version claimed)	0E80h
FC-PI-4 T11/1647-D revision 8.0	0E82h
FC-PLDA (no version claimed)	1340h
FC-PLDA ANSI INCITS TR-19 1998	135Ch
FC-PLDA T11/1162 revision 2.1	135Bh
FCP-2 (no version claimed)	0900h
FCP-2 ANSI INCITS 350-2003	0917h
FCP-2 T10/1144-D revision 8	0918h
FCP-2 T10/1144-D revision 4	0901h
FCP-2 T10/1144-D revision 7	0915h
FCP-2 T10/1144-D revision 7a	0916h
FCP-3 (no version claimed)	0A00h
FCP-3 ISO/IEC 14776-223	0A1Ch
FCP-3 ANSI INCITS 416-2006	0A11h
FCP-3 T10/1560-D revision 4	0A0Fh
FCP-3 T10/1560-D revision 3f	0A07h
Annex D contains the version descriptor value assignments in numeric order.	



**Table 135 — Version descriptor values** (part 4 of 9)

<b>Standard</b>	<b>Version Descriptor Value</b>
FCP-4 (no version claimed)	0A40h
FC-SP (no version claimed)	0E40h
FC-SP ANSI INCITS 426-2007	0E45h
FC-SP T11/1570-D revision 1.6	0E42h
FC-SP-2 (no version claimed)	0EC0h
FC-Tape (no version claimed)	1300h
FC-Tape ANSI INCITS TR-24 1999	131Ch
FC-Tape T11/1315 revision 1.17	131Bh
FC-Tape T11/1315 revision 1.16	1301h
IEEE 1394 (no version claimed)	14A0h
ANSI IEEE 1394-1995	14BDh
IEEE 1394a (no version claimed)	14C0h
IEEE 1394b (no version claimed)	14E0h
iSCSI (no version claimed)	0960h
MMC (no version claimed)	0140h
MMC ANSI INCITS 304-1997	015Ch
MMC T10/1048-D revision 10a	015Bh
MMC-2 (no version claimed)	0240h
MMC-2 ANSI INCITS 333-2000	025Ch
MMC-2 T10/1228-D revision 11a	025Bh
MMC-2 T10/1228-D revision 11	0255h
MMC-3 (no version claimed)	02A0h
MMC-3 ANSI INCITS 360-2002	02B8h
MMC-3 T10/1363-D revision 10g	02B6h
MMC-3 T10/1363-D revision 9	02B5h
MMC-4 (no version claimed)	03A0h
MMC-4 ANSI INCITS 401-2005	03BFh
MMC-4 T10/1545-D revision 5a	03B1h
MMC-4 T10/1545-D revision 3	03BDh
MMC-4 T10/1545-D revision 5	03B0h
MMC-5 (no version claimed)	0420h
MMC-5 T10/1675-D revision 04	0432h
MMC-5 ANSI INCITS 430-2007	0434h
MMC-5 T10/1675-D revision 03	042Fh
MMC-5 T10/1675-D revision 03b	0431h
MMC-6 (no version claimed)	04E0h
OCRW (no version claimed)	0280h
OCRW ISO/IEC 14776-381	029Eh
Annex D contains the version descriptor value assignments in numeric order.	

**Table 135 — Version descriptor values** (part 5 of 9)

<b>Standard</b>	<b>Version Descriptor Value</b>
OSD (no version claimed)	0340h
OSD ANSI INCITS 400-2004	0356h
OSD T10/1355-D revision 10	0355h
OSD T10/1355-D revision 0	0341h
OSD T10/1355-D revision 7a	0342h
OSD T10/1355-D revision 8	0343h
OSD T10/1355-D revision 9	0344h
OSD-2 (no version claimed)	0440h
OSD-2 T10/1729-D revision 4	0444h
RBC (no version claimed)	0220h
RBC ANSI INCITS 330-2000	023Ch
RBC T10/1240-D revision 10a	0238h
SAM (no version claimed)	0020h
SAM ANSI INCITS 270-1996	003Ch
SAM T10/0994-D revision 18	003Bh
SAM-2 (no version claimed)	0040h
SAM-2 ISO/IEC 14776-412	005Eh
SAM-2 ANSI INCITS 366-2003	005Ch
SAM-2 T10/1157-D revision 24	0055h
SAM-2 T10/1157-D revision 23	0054h
SAM-3 (no version claimed)	0060h
SAM-3 ANSI INCITS 402-2005	0077h
SAM-3 T10/1561-D revision 14	0076h
SAM-3 T10/1561-D revision 7	0062h
SAM-3 T10/1561-D revision 13	0075h
SAM-4 (no version claimed)	0080h
SAM-4 T10/1683-D revision 14	008Bh
SAM-4 T10/1683-D revision 13	0087h
SAM-5 (no version claimed)	00A0h
SAS (no version claimed)	0BE0h
SAS ANSI INCITS 376-2003	0BFDh
SAS T10/1562-D revision 05	0BFCCh
SAS T10/1562-D revision 01	0BE1h
SAS T10/1562-D revision 03	0BF5h
SAS T10/1562-D revision 04	0BFAh
SAS T10/1562-D revision 04	0BFBh
SAS-1.1 (no version claimed)	0C00h
SAS-1.1 ANSI INCITS 417-2006	0C11h
Annex D contains the version descriptor value assignments in numeric order.	

**Table 135 — Version descriptor values** (part 6 of 9)

<b>Standard</b>	<b>Version Descriptor Value</b>
SAS-1.1 T10/1601-D revision 10	0C0Fh
SAS-1.1 T10/1601-D revision 9	0C07h
SAS-2 (no version claimed)	0C20h
SAS-2 T10/1760-D revision 14	0C23h
SAT (no version claimed)	1EA0h
SAT ANSI INCITS 431-2007	1EADh
SAT T10/1711-D revision 9	1EABh
SAT T10/1711-D revision 8	1EA7h
SAT-2 (no version claimed)	1EC0h
SAT-2 T10/1826-D revision 6	1EC4h
SBC (no version claimed)	0180h
SBC ANSI INCITS 306-1998	019Ch
SBC T10/0996-D revision 08c	019Bh
SBC-2 (no version claimed)	0320h
SBC-2 ISO/IEC 14776-322	033Eh
SBC-2 ANSI INCITS 405-2005	033Dh
SBC-2 T10/1417-D revision 16	033Bh
SBC-2 T10/1417-D revision 5a	0322h
SBC-2 T10/1417-D revision 15	0324h
SBC-3 (no version claimed)	04C0h
SBP-2 (no version claimed)	08E0h
SBP-2 ANSI INCITS 325-1998	08FCh
SBP-2 T10/1155-D revision 04	08FBh
SBP-3 (no version claimed)	0980h
SBP-3 ANSI INCITS 375-2004	099Ch
SBP-3 T10/1467-D revision 5	099Bh
SBP-3 T10/1467-D revision 1f	0982h
SBP-3 T10/1467-D revision 3	0994h
SBP-3 T10/1467-D revision 4	099Ah
SCC (no version claimed)	0160h
SCC ANSI INCITS 276-1997	017Ch
SCC T10/1047-D revision 06c	017Bh
SCC-2 (no version claimed)	01E0h
SCC-2 ANSI INCITS 318-1998	01FCh
SCC-2 T10/1125-D revision 04	01FBh
SES (no version claimed)	01C0h
SES ANSI INCITS 305-1998	01DCh
SES T10/1212-D revision 08b	01DBh
Annex D contains the version descriptor value assignments in numeric order.	

**Table 135 — Version descriptor values** (part 7 of 9)

<b>Standard</b>	<b>Version Descriptor Value</b>
SES ANSI INCITS 305-1998 w/ Amendment ANSI INCITS.305/AM1-2000	01DEh
SES T10/1212 revision 08b w/ Amendment ANSI INCITS.305/AM1-2000	01DDh
SES-2 (no version claimed)	03E0h
SES-2 T10/1559-D revision 20	03EBh
SES-2 T10/1559-D revision 16	03E1h
SES-2 T10/1559-D revision 19	03E7h
SIP (no version claimed)	08A0h
SIP ANSI INCITS 292-1997	08BCh
SIP T10/0856-D revision 10	08BBh
SMC (no version claimed)	01A0h
SMC ISO/IEC 14776-351	01BEh
SMC ANSI INCITS 314-1998	01BCh
SMC T10/0999-D revision 10a	01BBh
SMC-2 (no version claimed)	02E0h
SMC-2 ANSI INCITS 382-2004	02FEh
SMC-2 T10/1383-D revision 7	02FDh
SMC-2 T10/1383-D revision 5	02F5h
SMC-2 T10/1383-D revision 6	02FCh
SMC-3 (no version claimed)	0480h
SPC (no version claimed)	0120h
SPC ANSI INCITS 301-1997	013Ch
SPC T10/0995-D revision 11a	013Bh
SPC-2 (no version claimed)	0260h
SPC-2 ISO/IEC 14776-452	0278h
SPC-2 ANSI INCITS 351-2001	0277h
SPC-2 T10/1236-D revision 20	0276h
SPC-2 T10/1236-D revision 12	0267h
SPC-2 T10/1236-D revision 18	0269h
SPC-2 T10/1236-D revision 19	0275h
SPC-3 (no version claimed)	0300h
SPC-3 ANSI INCITS 408-2005	0314h
SPC-3 T10/1416-D revision 23	0312h
SPC-3 T10/1416-D revision 7	0301h
SPC-3 T10/1416-D revision 21	0307h
SPC-3 T10/1416-D revision 22	030Fh
SPC-4 (no version claimed)	0460h
SPC-4 T10/1731-D revision 16	0461h
Annex D contains the version descriptor value assignments in numeric order.	

**Table 135 — Version descriptor values** (part 8 of 9)

<b>Standard</b>	<b>Version Descriptor Value</b>
SPI (no version claimed)	0AA0h
SPI ANSI INCITS 253-1995	0ABAh
SPI T10/0855-D revision 15a	0AB9h
SPI ANSI INCITS 253-1995 with SPI Amnd ANSI INCITS 253/AM1-1998	0ABCh
SPI T10/0855-D revision 15a with SPI Amnd revision 3a	0ABBh
SPI-2 (no version claimed)	0AE0h
SPI-2 ANSI INCITS 302-1999	0AFCh
SPI-2 T10/1142-D revision 20b	0AFBh
SPI-3 (no version claimed)	0B00h
SPI-3 ANSI INCITS 336-2000	0B1Ch
SPI-3 T10/1302-D revision 14	0B1Ah
SPI-3 T10/1302-D revision 10	0B18h
SPI-3 T10/1302-D revision 13a	0B19h
SPI-4 (no version claimed)	0B40h
SPI-4 ANSI INCITS 362-2002	0B56h
SPI-4 T10/1365-D revision 7	0B54h
SPI-4 T10/1365-D revision 9	0B55h
SPI-4 T10/1365-D revision 10	0B59h
SPI-5 (no version claimed)	0B60h
SPI-5 ANSI INCITS 367-2003	0B7Ch
SPI-5 T10/1525-D revision 6	0B7Bh
SPI-5 T10/1525-D revision 3	0B79h
SPI-5 T10/1525-D revision 5	0B7Ah
SRP (no version claimed)	0940h
SRP ANSI INCITS 365-2002	095Ch
SRP T10/1415-D revision 16a	0955h
SRP T10/1415-D revision 10	0954h
SSA-PH2 (no version claimed)	1360h
SSA-PH2 ANSI INCITS 293-1996	137Ch
SSA-PH2 T10.1/1145-D revision 09c	137Bh
SSA-PH3 (no version claimed)	1380h
SSA-PH3 ANSI INCITS 307-1998	139Ch
SSA-PH3 T10.1/1146-D revision 05b	139Bh
SSA-S2P (no version claimed)	0880h
SSA-S2P ANSI INCITS 294-1996	089Ch
SSA-S2P T10.1/1121-D revision 07b	089Bh
SSA-S3P (no version claimed)	0860h
SSA-S3P ANSI INCITS 309-1998	087Ch
Annex D contains the version descriptor value assignments in numeric order.	

**Table 135 — Version descriptor values** (part 9 of 9)

<b>Standard</b>	<b>Version Descriptor Value</b>
SSA-S3P T10.1/1051-D revision 05b	087Bh
SSA-TL1 (no version claimed)	0840h
SSA-TL1 ANSI INCITS 295-1996	085Ch
SSA-TL1 T10.1/0989-D revision 10b	085Bh
SSA-TL2 (no version claimed)	0820h
SSA-TL2 ANSI INCITS 308-1998	083Ch
SSA-TL2 T10.1/1147-D revision 05b	083Bh
SSC (no version claimed)	0200h
SSC ANSI INCITS 335-2000	021Ch
SSC T10/0997-D revision 22	0207h
SSC T10/0997-D revision 17	0201h
SSC-2 (no version claimed)	0360h
SSC-2 ANSI INCITS 380-2003	037Dh
SSC-2 T10/1434-D revision 9	0375h
SSC-2 T10/1434-D revision 7	0374h
SSC-3 (no version claimed)	0400h
SSC-3 T10/1611-D revision 04a	0403h
SST (no version claimed)	0920h
SST T10/1380-D revision 8b	0935h
UAS (no version claimed)	1740h
Universal Serial Bus Specification, Revision 1.1	1728h
Universal Serial Bus Specification, Revision 2.0	1729h
USB Mass Storage Class Bulk-Only Transport, Revision 1.0	1730h
Version Descriptor Not Supported or No Standard Identified	0000h
Annex D contains the version descriptor value assignments in numeric order.	

### 6.4.3 SCSI Parallel Interface specific INQUIRY data

Portions of bytes 6 and 7 and all of byte 56 of the standard INQUIRY data shall be used only by SCSI target devices that implement the SCSI Parallel Interface. These fields are noted in table 130. For details on how the SPI-specific fields relate to the SCSI Parallel Interface see SPI-n (where n is 2 or greater). Table 136 shows just the SPI-specific standard INQUIRY fields. The definitions of the SCSI Parallel Interface specific fields shall be as follows.

**Table 136 — SPI-specific standard INQUIRY bits**

Bit Byte	7	6	5	4	3	2	1	0
6	see table 130							ADDR16
7	see table 130		WBUS16	SYNC	see table 130	Obsolete	see table 130	
	⋮							
56	Reserved				CLOCKING		QAS	IUS

A wide SCSI address 16 (ADDR16) bit of one indicates that the SCSI target device supports 16-bit wide SCSI addresses. A value of zero indicates that the SCSI target device does not support 16-bit wide SCSI addresses.

A wide bus 16 (WBUS16) bit of one indicates that the SCSI target device supports 16-bit wide data transfers. A value of zero indicates that the SCSI target device does not support 16-bit wide data transfers.

A synchronous transfer (SYNC) bit of one indicates that the SCSI target device supports synchronous data transfer. A value of zero indicates the SCSI target device does not support synchronous data transfer.

The obsolete bit 2 in byte 7 indicates whether the SCSI target device supports an obsolete data transfers management mechanism defined in SPI-2.

Table 137 defines the relationships between the ADDR16 and WBUS16 bits.

**Table 137 — Maximum logical device configuration table**

ADDR16	WBUS16	Description
0	0	8 bit wide data path on a single cable with 8 SCSI IDs supported
0	1	16 bit wide data path on a single cable with 8 SCSI IDs supported
1	1	16 bit wide data path on a single cable with 16 SCSI IDs supported

The CLOCKING field shall not apply to asynchronous transfers and is defined in table 138.

**Table 138 — CLOCKING field**

Code	Description
00b	Indicates the target port supports only ST
01b	Indicates the target port supports only DT
10b	Reserved
11b	Indicates the target port supports ST and DT

A quick arbitration and selection supported (QAS) bit of one indicates that the target port supports quick arbitration and selection. A value of zero indicates that the target port does not support quick arbitration and selection.

An information units supported (IUS) bit of one indicates that the SCSI target device supports information unit transfers. A value of zero indicates that the SCSI target device does not support information unit transfers.

NOTE 29 - The acronyms ST and DT and the terms 'quick arbitration and selection' and 'information units' are defined in SPI-5.

#### 6.4.4 Vital product data

The application client requests the vital product data information by setting the EVPD bit to one and specifying the page code of a vital product data. See 7.7 for details about vital product data. The information returned consists of configuration data (e.g., vendor identification, product identification, model, serial number), manufacturing data (e.g., plant and date of manufacture), field replaceable unit data and other vendor specific or device specific data. If the device server does not implement the requested page, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The device server should have the ability to process the INQUIRY command even when an error occurs that prohibits normal command completion. In such a case, CHECK CONDITION status should be returned for commands other than INQUIRY or REQUEST SENSE. The sense data returned may contain the field replaceable unit code. The vital product data may be obtained for the failing device using the INQUIRY command.

This standard defines a format that allows device-independent application client software to display the vital product data returned by the INQUIRY command. The contents of the data may be vendor specific, and may be unusable without detailed information about the device.

This standard does not define the location or method of storing the vital product data. The retrieval of the data may require completion of initialization operations within the device, that may induce delays before the data is available to the application client. Time-critical requirements are an implementation consideration and are not addressed in this standard.



## 6.5 LOG SELECT command

### 6.5.1 Introduction

The LOG SELECT command (see table 139) provides a means for an application client to manage statistical information maintained by the SCSI target device about the SCSI target device or its logical units. Device servers that implement the LOG SELECT command shall also implement the LOG SENSE command. Structures in the form of log parameters within log pages are defined as a way to manage the log data. The LOG SELECT command provides a method for sending zero or more log pages via the Data-Out Buffer. This standard defines the format of the log pages (see 7.2) and defines some of the conditions and events that are logged.

**Table 139 — LOG SELECT command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (4Ch)							
1	Reserved						PCR	SP
2	PC		PAGE CODE					
3	SUBPAGE CODE							
4	Reserved							
6								
7	(MSB)							
8	PARAMETER LIST LENGTH						(LSB)	
9	CONTROL							

The parameter code reset (PCR) bit instructs a device server whether or not to set parameters to their vendor specific default values (e.g., zero) as described in table 142.

The save parameters (SP) bit instructs a device server whether or not to save parameters to non-volatile memory as described in table 142.

The page control (PC) field specifies which data counter parameter values (i.e., when the FORMAT AND LINKING field (see 7.2.1) contains 00b or 10b) shall be processed by a device server in response to a LOG SELECT command as described in table 140. The PC field shall be ignored for list parameters (i.e., when the FORMAT AND LINKING field contains 01b or 11b).

**Table 140 — Page control (pc) field**

Value	Description
00b	Threshold values
01b	Cumulative values
10b	Default threshold values
11b	Default cumulative values
<sup>a</sup> The threshold values and cumulative values for data counter parameters are: <ol style="list-style-type: none"> <li>1) The current values if there has been an update to a cumulative parameter value (e.g., by a LOG SELECT command or by a device specific event) in the specified page or pages since the last logical unit reset occurred;</li> <li>2) The saved values, if saved parameters are implemented, current values have been saved, and an update has not occurred since the last logical unit reset; or</li> <li>3) The vendor specific default values, if saved values are not available or not implemented.</li> </ol>	

When evaluated together, the combination of the values in the PCR bit, the SP bit, and the PC field specify the actions that a SCSI target device performs while processing a LOG SELECT command.

If the PARAMETER LIST LENGTH field is set to zero, the PAGE CODE field and SUBPAGE CODE field specify the log page or log pages to which the other CDB fields apply (see 6.5.2).

Since each log page in the parameter list contains a PAGE CODE field and SUBPAGE CODE field (see 7.2.1.1), the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB, if:

- a) The PARAMETER LIST LENGTH field contains a value other than zero, and:
  - A) The PAGE CODE field contains a value other than zero; or
  - B) the SUBPAGE CODE field contains a value other than zero.

The PARAMETER LIST LENGTH field specifies the length in bytes of the parameter list that shall be located in the Data-Out Buffer.

If the PARAMETER LIST LENGTH field contains a value other than zero, the actions that a SCSI target device performs after receiving a LOG SELECT command are determined by the values in the PCR bit, the SP bit, and the PC field as described in table 283 (see 7.2.1.1).

If the PARAMETER LIST LENGTH field contains zero, no log pages shall be transferred. This condition shall not be considered an error. The LOG SELECT command shall be processed as described in 6.5.2.

### 6.5.2 Processing LOG SELECT when the parameter list length is zero

If the PARAMETER LIST LENGTH field is set to zero (i.e., when there is no parameter data being sent with a LOG SELECT command), the SCSI target device responds by processing the log parameter values as described in this subclause.

The PAGE CODE field and SUBPAGE CODE field (see table 141) specify the log page or log pages to which the other CDB fields apply (see table 142).

**Table 141 — PAGE CODE field and SUBPAGE CODE field**

PAGE CODE field	SUBPAGE CODE field	Description
00h	00h	All log parameters in all log pages <sup>a</sup>
00h to 3Fh	01h to FEh	All log parameters in the log page specified by the page code and subpage code
00h to 3Fh	FFh	All log parameters in the log pages specified by page code and all subpage codes
01h to 3Fh	00h	All log parameters in the log page specified by the page code
<sup>a</sup> This is equivalent to the LOG SELECT command operation specified by previous versions of this standard.		

Table 142 defines the meaning of the combinations of values for the PCR bit, the SP bit, and the PC field.

**Table 142 — PCR bit, SP bit, and PC field meanings when parameter list length is zero (part 1 of 3)**

PCR bit	SP bit	PC field	Description
0b	0b	0xb	This is not an error. The device server shall make no changes to any log parameter values and shall not save any values to non-volatile media.
0b	1b	00b	The device server shall make no changes to any log parameter values and shall process the optional saving of current parameter values as follows: <ul style="list-style-type: none"> <li>a) If the values are current threshold data counter parameters, then:               <ul style="list-style-type: none"> <li>A) If the device server implements saving of the current threshold values, the device server shall save all current threshold values to non-volatile media; or</li> <li>B) If the device server does not implement saving of the current threshold values, the device server shall terminate the command<sup>a</sup>.</li> </ul> </li> <li>or</li> <li>b) If the values are current list parameters, then:               <ul style="list-style-type: none"> <li>A) If the device server implements saving of the current list parameters, the device server shall save all current list parameters to non-volatile media; or</li> <li>B) If the device server does not implement saving of the current list parameters, the device server shall terminate the command<sup>a</sup>.</li> </ul> </li> </ul>
<sup>a</sup> The command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.			
<sup>b</sup> Vendor specific default threshold values and vendor specific default cumulative values may be zero.			

**Table 142 — PCR bit, SP bit, and PC field meanings when parameter list length is zero (part 2 of 3)**

PCR bit	SP bit	PC field	Description
0b	1b	01b	<p>The device server shall make no change to any log parameter values and shall process the optional saving of current parameter values as follows:</p> <p>a) If the values are current cumulative data counter parameters, then:</p> <p>A) If the device server implements saving of the current cumulative values, the device server shall save all current cumulative values to non-volatile media; or</p> <p>B) If the device server does not implement saving of the current cumulative values, the device server shall terminate the command<sup>a</sup>.</p> <p>or</p> <p>b) If the values are current list parameters, then:</p> <p>A) If the device server implements saving of the current list parameters, the device server shall save all current list parameters to non-volatile media; or</p> <p>B) If the device server does not implement saving of the current list parameters, the device server shall terminate the command<sup>a</sup>.</p>
0b	xb	10b	The device server shall set all current threshold values to the vendor specific default threshold values <sup>b</sup> and shall not save any values to non-volatile media.
0b	xb	11b	The device server shall set all current cumulative values to the vendor specific default cumulative values <sup>b</sup> and shall not save any values to non-volatile media.
1b	0b	xxb	<p>The device server shall:</p> <ol style="list-style-type: none"> <li>1) Set all current threshold values to the vendor specific default threshold values<sup>b</sup>;</li> <li>2) Set all current cumulative values to the vendor specific default cumulative values<sup>b</sup>;</li> <li>3) Set all list parameters to their vendor specific default values; and</li> <li>4) Not save any values to non-volatile media.</li> </ol>
1b	1b	00b	<p>The device server shall process the optional saving of current threshold values as follows:</p> <p>a) If the device server implements saving of the current threshold values, the device server shall:</p> <ol style="list-style-type: none"> <li>1) Save all current threshold values to non-volatile media;</li> <li>2) Set all current threshold values to the vendor specific default threshold values<sup>b</sup>;</li> <li>3) Set all current cumulative values to the vendor specific default cumulative values<sup>b</sup>, and</li> <li>4) Set all list parameters to their vendor specific default values.</li> </ol> <p>or</p> <p>b) If the device server does not implement saving of the current threshold values, the device server shall terminate the command<sup>a</sup>.</p>
<p><sup>a</sup> The command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.</p> <p><sup>b</sup> Vendor specific default threshold values and vendor specific default cumulative values may be zero.</p>			

**Table 142 — PCR bit, SP bit, and PC field meanings when parameter list length is zero (part 3 of 3)**

PCR bit	SP bit	PC field	Description
1b	1b	01b	<p>The device server shall process the optional saving of current cumulative values as follows:</p> <p>a) If the device server implements saving of the current cumulative values, the device server shall:</p> <ol style="list-style-type: none"> <li>1) Save all current cumulative values to non-volatile media;</li> <li>2) Set all current threshold values to the vendor specific default threshold values<sup>b</sup>;</li> <li>3) Set all current cumulative values to the vendor specific default cumulative values<sup>b</sup>, and</li> <li>4) Set all list parameters to their vendor specific default values.</li> </ol> <p>or</p> <p>b) If the device server does not implement saving of the current cumulative values, the device server shall terminate the command<sup>a</sup>.</p>
1b	1b	1xb	<p>The device server shall:</p> <ol style="list-style-type: none"> <li>1) Set all current threshold values to the vendor specific default threshold values<sup>b</sup>;</li> <li>2) Set all current cumulative values to the vendor specific default cumulative values<sup>b</sup>;</li> <li>3) Set all list parameters to their vendor specific default values; and</li> <li>4) Not save any values to non-volatile media.</li> </ol>
<p><sup>a</sup> The command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.</p> <p><sup>b</sup> Vendor specific default threshold values and vendor specific default cumulative values may be zero.</p>			

The current cumulative values may be updated by the device server as defined for the specific log page or by the application client using the LOG SELECT command. The current threshold values may only be modified by the application client via the LOG SELECT command.

NOTE 30 - Log pages or log parameters that are not available may become available at some later time (e.g., after the logical unit has become ready).

## 6.6 LOG SENSE command

The LOG SENSE command (see table 143) provides a means for the application client to retrieve statistical or other operational information maintained by the SCSI target device about the SCSI target device or its logical units. It is a complementary command to the LOG SELECT command.

**Table 143 — LOG SENSE command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (4Dh)							
1	Reserved						PPC	SP
2	PC		PAGE CODE					
3	SUBPAGE CODE							
4	Reserved							
5	(MSB) _____							
6	PARAMETER POINTER _____ (LSB)							
7	(MSB) _____							
8	ALLOCATION LENGTH _____ (LSB)							
9	CONTROL							

The parameter pointer control (PPC) bit controls the type of parameters requested from the device server:

- A PPC bit set to one specifies that the device server shall return a log page with parameter code values that have changed since the last LOG SELECT or LOG SENSE command. The device server shall return only those parameter codes that are greater than or equal to the contents of the PARAMETER POINTER field in ascending order of parameter codes from the specified log page;
- A PPC bit set to zero specifies that the device server shall return those parameter codes that are greater than or equal to the contents of the PARAMETER POINTER field in ascending order of parameter codes from the specified log page; and
- A PPC bit set to zero and a PARAMETER POINTER field set to zero specifies that the device server shall return all available log parameters from the specified log page.

If the PPC bit is set to one and the value of the SUBPAGE CODE field is set to FFh then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

Saving parameters is an optional function of the LOG SENSE command. If the logical unit does not implement saving log parameters and if the save parameters (SP) bit is set to one, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

An SP bit set to zero specifies the device server shall perform the specified LOG SENSE command and shall not save any log parameters. If saving log parameters is implemented, an SP bit set to one specifies that the device server shall perform the specified LOG SENSE command and shall save all log parameters identified as saveable by the DS bit (see 7.2) to a nonvolatile, vendor specific location.

For data counter log parameters (i.e., when the FORMAT AND LINKING field in the parameter control byte in the log parameter structure (see 7.2.1.2) contains 00b or 10b), the page control (PC) field (see table 140 in 6.5.1) specifies which log parameter values are to be returned by a device server in response to a LOG SENSE command.

For list parameters (i.e., when the FORMAT AND LINKING field in the parameter control byte in the log parameter structure (see 7.2.1.2) contains 01b or 11b), the PC field shall be ignored. If the parameters specified by the PAGE CODE field and SUBPAGE CODE field in the CDB are list parameters, then the parameter values returned by a device server in response to a LOG SENSE command are determined as follows:

- 1) The current list parameter values, if there has been an update to a list parameter value (e.g., by a LOG SELECT command or by a device specific event) in the specified page or pages since the last logical unit reset occurred;
- 2) The saved list parameter values, if saved parameters are implemented and an update has not occurred since the last logical unit reset; or
- 3) The vendor specific default list parameter values, if saved values are not available or not implemented and an update has not occurred since the last logical unit reset.

The PAGE CODE field and SUBPAGE CODE field specify which log page of data is being requested (see 7.2). If the log page specified by the page code and subpage code combination is reserved or not implemented, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The PARAMETER POINTER field allows the application client to request parameter data beginning from a specific parameter code to the maximum allocation length or the maximum parameter code supported by the logical unit, whichever is less. If the value of the PARAMETER POINTER field is larger than the largest available parameter code known to the device server for the specified log page, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The ALLOCATION LENGTH field is defined in 4.3.5.6.

Log parameters within the specified log page shall be transferred in ascending order according to parameter code.

## 6.7 MANAGEMENT PROTOCOL IN command

### 6.7.1 MANAGEMENT PROTOCOL IN command description

The MANAGEMENT PROTOCOL IN command (see table 144) is used to retrieve management protocol information (see 6.7.2) or the results of one or more MANAGEMENT PROTOCOL OUT commands (see 6.8).

**Table 144 — MANAGEMENT PROTOCOL IN command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A3h)							
1	Reserved			SERVICE ACTION (10h)				
2	MANAGEMENT PROTOCOL							
3	MANAGEMENT PROTOCOL SPECIFIC1							
5								
6	(MSB)	ALLOCATION LENGTH						
9								
10	MANAGEMENT PROTOCOL SPECIFIC2							
11	CONTROL							

The MANAGEMENT PROTOCOL field (see table 145) specifies which management protocol is being used.

**Table 145 — MANAGEMENT PROTOCOL field in MANAGEMENT PROTOCOL IN command**

Code	Description	Reference
00h	Management protocol information	6.7.2
01h to 2Fh	Reserved	3.1.165
30h to 35h	Defined by the SNIA	
36h to EFh	Reserved	
F0h to FFh	Vendor Specific	

The contents of the MANAGEMENT PROTOCOL SPECIFIC1 field and MANAGEMENT PROTOCOL SPECIFIC2 field depend on the protocol specified by the MANAGEMENT PROTOCOL field (see table 145).

The ALLOCATION LENGTH field is defined in 4.3.5.6.

Indications of data overrun or underrun and the mechanism, if any, for processing retries depend on the protocol specified by the MANAGEMENT PROTOCOL field (see table 145).

Any association between a previous MANAGEMENT PROTOCOL OUT command and the data transferred by a MANAGEMENT PROTOCOL IN command depends on the protocol specified by the MANAGEMENT PROTOCOL field (see table 145). If the device server has no data to transfer (e.g., the results for any previous MANAGEMENT PROTOCOL OUT commands are not yet available), the device server may transfer data indicating it has no other data to transfer.

The format of the data transferred depends on the protocol specified by the MANAGEMENT PROTOCOL field (see table 145).



The device server shall retain data resulting from a MANAGEMENT PROTOCOL OUT command, if any, until one of the following events is processed:

- a) Transfer of the data via a MANAGEMENT PROTOCOL IN command from the same I\_T\_L nexus as defined by the protocol specified by the MANAGEMENT PROTOCOL field (see table 145);
- b) Logical unit reset (See SAM-4); or
- c) I\_T nexus loss (See SAM-4) associated with the I\_T nexus that sent the MANAGEMENT PROTOCOL OUT command.

If the data is lost due to one of these events the application client may send a new MANAGEMENT PROTOCOL OUT command to retry the operation.

## 6.7.2 Management protocol information description

### 6.7.2.1 Overview

The purpose of the management protocol information management protocol (i.e., the MANAGEMENT PROTOCOL field set to 00h in a MANAGEMENT PROTOCOL IN command) is to transfer management protocol related information from the logical unit. A MANAGEMENT PROTOCOL IN command in which the MANAGEMENT PROTOCOL field is set to 00h is not associated with any previous MANAGEMENT PROTOCOL OUT command and shall be processed without regard for whether a MANAGEMENT PROTOCOL OUT command has been processed.

If the MANAGEMENT PROTOCOL IN command is supported, the MANAGEMENT PROTOCOL value of 00h shall be supported as defined in this standard.

### 6.7.2.2 CDB description

When the MANAGEMENT PROTOCOL field is set to 00h in a MANAGEMENT PROTOCOL IN command, the MANAGEMENT PROTOCOL SPECIFIC1 field is reserved and the MANAGEMENT PROTOCOL SPECIFIC2 field (see table 146) contains a single numeric value as defined in 3.5.

**Table 146 — MANAGEMENT PROTOCOL SPECIFIC2 field for MANAGEMENT PROTOCOL IN protocol 00h**

Code	Description	Support	Reference
00h	Supported management protocol list	Mandatory	6.7.2.3
01h to FFh	Reserved		

All other CDB fields for MANAGEMENT PROTOCOL IN command shall meet the requirements stated in 6.7.1.

Each time a MANAGEMENT PROTOCOL IN command with the MANAGEMENT PROTOCOL field set to 00h is received, the device server shall transfer the data defined 6.7.2.3 starting with byte 0.

### 6.7.2.3 Supported management protocols list description

If the MANAGEMENT PROTOCOL field is set to 00h and the MANAGEMENT PROTOCOL SPECIFIC2 field is set to 00h in a MANAGEMENT PROTOCOL IN command, then the parameter data shall have the format shown in table 147.

**Table 147 — Supported management protocols MANAGEMENT PROTOCOL IN parameter data**

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved							
5								
6	(MSB)	SUPPORTED MANAGEMENT PROTOCOL LIST LENGTH						
7		(n-7)						(LSB)
Supported management protocol list								
8	SUPPORTED MANAGEMENT PROTOCOL [first] (00h)							
	⋮							
n	SUPPORTED MANAGEMENT PROTOCOL [last]							

The SUPPORTED MANAGEMENT PROTOCOL LIST LENGTH field indicates the total length, in bytes, of the supported management protocol list that follows.

Each SUPPORTED MANAGEMENT PROTOCOL field in the supported management protocols list shall contain one of the management protocol values supported by the logical unit. The values shall be listed in ascending order starting with 00h.

The total data length shall conform to the ALLOCATION LENGTH field requirements (see 4.3.5.6).

## 6.8 MANAGEMENT PROTOCOL OUT command

The MANAGEMENT PROTOCOL OUT command (see table 148) is used to send data to the logical unit. The data sent specifies one or more operations to be performed by the logical unit. The format and function of the operations depends on the contents of the MANAGEMENT PROTOCOL field (see table 148). Depending on the protocol specified by the MANAGEMENT PROTOCOL field, the application client may use the MANAGEMENT PROTOCOL IN command (see 6.7) to retrieve data derived from these operations.

**Table 148 — MANAGEMENT PROTOCOL OUT command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A4h)							
1	Reserved			SERVICE ACTION (10h)				
2	MANAGEMENT PROTOCOL							
3	MANAGEMENT PROTOCOL SPECIFIC1							
5								
6	(MSB)	PARAMETER LIST LENGTH						(LSB)
9								
10	MANAGEMENT PROTOCOL SPECIFIC2							
11	CONTROL							

The MANAGEMENT PROTOCOL field (see table 149) specifies which management protocol is being used.

**Table 149 — MANAGEMENT PROTOCOL field in MANAGEMENT PROTOCOL OUT command**

Code	Description	Reference
00h to 2Fh	Reserved	3.1.165
30h to 35h	Defined by the SNIA	
36h to EFh	Reserved	
F0h to FFh	Vendor Specific	

The contents of the MANAGEMENT PROTOCOL SPECIFIC1 field and MANAGEMENT PROTOCOL SPECIFIC2 field depend on the protocol specified by the MANAGEMENT PROTOCOL field (see table 149).

The PARAMETER LIST LENGTH field is defined in 4.3.5.5.

Any association between a MANAGEMENT PROTOCOL OUT command and a subsequent MANAGEMENT PROTOCOL IN command depends on the protocol specified by the MANAGEMENT PROTOCOL field (see table 149). Each management protocol shall define whether:

- The device server shall complete the command with GOOD status as soon as it determines the data has been correctly received. An indication that the data has been processed is obtained by sending a MANAGEMENT PROTOCOL IN command and receiving the results in the associated data transfer; or
- The device server shall complete the command with GOOD status only after the data has been successfully processed and an associated MANAGEMENT PROTOCOL IN command is not required.

The format of the data transferred depends on the protocol specified by the MANAGEMENT PROTOCOL field (see table 149).

## 6.9 MODE SELECT(6) command

The MODE SELECT(6) command (see table 150) provides a means for the application client to specify medium, logical unit, or peripheral device parameters to the device server. Device servers that implement the MODE SELECT(6) command shall also implement the MODE SENSE(6) command. Application clients should issue MODE SENSE(6) prior to each MODE SELECT(6) to determine supported mode pages, page lengths, and other parameters.

**Table 150 — MODE SELECT(6) command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (15h)							
1	Reserved			PF	Reserved			SP
2	Reserved							
3								
4	PARAMETER LIST LENGTH							
5	CONTROL							

Logical units shall share mode parameter header and block descriptor values across all I\_T nexuses. I\_T nexus loss shall not affect mode parameter header, block descriptor, and mode page values.

Logical units shall maintain current and saved values of each mode page based on any of the policies listed in table 151. The mode page policy used for each mode page may be reported in the Mode Page Policy VPD page (see 7.7.6).

**Table 151 — Mode page policies**

Mode page policy	Number of mode page copies
Shared	One copy of the mode page that is shared by all I_T nexuses.
Per target port	A separate copy of the mode page for each target port with each copy shared by all initiator ports.
Per I_T nexus	A separate copy of the mode page for each I_T nexus

After a logical unit reset, each mode parameter header, block descriptor, and mode page shall revert to saved values if supported or default values if saved values are not supported.

If an application client sends a MODE SELECT command that changes any parameters applying to other I\_T nexuses, the device server shall establish a unit attention condition (see SAM-4) for the initiator port associated with every I\_T nexus except the I\_T nexus on which the MODE SELECT command was received, with the additional sense code set to MODE PARAMETERS CHANGED.

A page format (PF) bit set to zero specifies that all parameters after the block descriptors are vendor specific. A PF bit set to one specifies that the MODE SELECT parameters following the header and block descriptor(s) are structured as pages of related parameters and are as defined in this standard.

A save pages (SP) bit set to zero specifies that the device server shall perform the specified MODE SELECT operation, and shall not save any mode pages. If the logical unit implements no distinction between current and saved mode pages and the SP bit is set to zero, the command shall be terminated with CHECK CONDITION status,

with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB. An SP bit set to one specifies that the device server shall perform the specified MODE SELECT operation, and shall save to a nonvolatile vendor specific location all the saveable mode pages including any sent in the Data-Out Buffer. Mode pages that are saved are specified by the parameter saveable (PS) bit that is returned in the first byte of each mode page by the MODE SENSE command (see 7.4). If the PS bit is set to one in the MODE SENSE data, then the mode page shall be saveable by issuing a MODE SELECT command with the SP bit set to one. If the logical unit does not implement saved mode pages and the SP bit is set to one, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The PARAMETER LIST LENGTH field specifies the length in bytes of the mode parameter list that shall be contained in the Data-Out Buffer. A parameter list length of zero specifies that the Data-Out Buffer shall be empty. This condition shall not be considered as an error.

If the parameter list length results in the truncation of any mode parameter header, mode parameter block descriptor(s), or mode page, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

The mode parameter list for the MODE SELECT and MODE SENSE commands is defined in 7.4. Parts of each mode parameter list are defined in a device-type dependent manner. Definitions for the parts of each mode parameter list that are unique for each device-type may be found in the applicable command standards (see 3.1.27).

The device server shall terminate the MODE SELECT command with CHECK CONDITION status, set the sense key to ILLEGAL REQUEST, set the additional sense code to INVALID FIELD IN PARAMETER LIST, and shall not change any mode parameters in response to any of the following conditions:

- a) If the application client sets any field that is reported as not changeable by the device server to a value other than its current value;
- b) If the application client sets any field in the mode parameter header or block descriptor(s) to an unsupported value;
- c) If an application client sends a mode page with a page length not equal to the page length returned by the MODE SENSE command for that mode page;
- d) If the application client sends an unsupported value for a mode parameter and rounding is not implemented for that mode parameter; or
- e) If the application client sets any reserved field in the mode parameter list to a non-zero value and the device server checks reserved fields.

If the application client sends a value for a mode parameter that is outside the range supported by the device server and rounding is implemented for that mode parameter, the device server handles the condition by either:

- a) Rounding the parameter to an acceptable value and terminating the command as described in 5.4; or
- b) Terminating the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

A device server may alter any mode parameter in any mode page, even those reported as non-changeable, as a result of changes to other mode parameters.

The device server validates the non-changeable mode parameters against the current values that existed for those mode parameters prior to the MODE SELECT command.

NOTE 31 - The current values calculated by the device server may affect the application client's operation. The application client may issue a MODE SENSE command after each MODE SELECT command, to determine the current values.

## 6.10 MODE SELECT(10) command

The MODE SELECT(10) command (see table 152) provides a means for the application client to specify medium, logical unit, or peripheral device parameters to the device server. See the MODE SELECT(6) command (6.9) for a description of the fields and operation of this command. Application clients should issue MODE SENSE(10) prior to each MODE SELECT(10) to determine supported mode pages, page lengths, and other parameters. Device servers that implement the MODE SELECT(10) command shall also implement the MODE SENSE(10) command.

**Table 152 — MODE SELECT(10) command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (55h)							
1	Reserved			PF	Reserved			SP
2	Reserved							
6								
7	(MSB)							
8	PARAMETER LIST LENGTH							(LSB)
9	CONTROL							

## 6.11 MODE SENSE(6) command

### 6.11.1 MODE SENSE(6) command introduction

The MODE SENSE(6) command (see table 153) provides a means for a device server to report parameters to an application client. It is a complementary command to the MODE SELECT(6) command. Device servers that implement the MODE SENSE(6) command shall also implement the MODE SELECT(6) command.

**Table 153 — MODE SENSE(6) command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (1Ah)							
1	Reserved				DBD	Reserved		
2	PC		PAGE CODE					
3	SUBPAGE CODE							
4	ALLOCATION LENGTH							
5	CONTROL							

A disable block descriptors (DBD) bit set to zero specifies that the device server may return zero or more block descriptors in the returned MODE SENSE data (see 7.4). A DBD bit set to one specifies that the device server shall not return any block descriptors in the returned MODE SENSE data.

The page control (PC) field specifies the type of mode parameter values to be returned in the mode pages. The PC field is defined in table 154.

**Table 154 — Page control (pc) field**

Code	Type of parameter	Reference
00b	Current values	6.11.2
01b	Changeable values	6.11.3
10b	Default values	6.11.4
11b	Saved values	6.11.5

The PC field only affects the mode parameters within the mode pages, however the PS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field should return current values (i.e., as if PC is set to 00b). The mode parameter header and mode parameter block descriptor should return current values.

Some SCSI target devices may not distinguish between current and saved mode parameters and report identical values in response to a PC field of either 00b or 11b. See also the description of the save pages (SP) bit in the MODE SELECT command.

The PAGE CODE and SUBPAGE CODE fields (see table 155) specify which mode pages and subpages to return.

**Table 155 — Mode page code usage for all devices**

Page Code	Subpage Code	Description
00h	vendor specific	Vendor specific (does not require page format)
01h to 1Fh	00h	See specific device types (page_0 format)
	01h to DFh	See specific device types (sub_page format)
	E0h to FEh	Vendor specific (sub_page format)
	FFh	Return all subpages for the specified device specific mode page in the page_0 format for subpage 00h and in the sub_page format for subpages 01h to FEh
20h to 3Eh	00h	Vendor specific (page_0 format required)
	01h to FEh	Vendor specific (sub_page format required)
	FFh	Return all subpages for the specified vendor specific mode page in the page_0 format for subpage 00h and in the sub_page format for subpages 01h to FEh
3Fh	00h	Return all subpage 00h mode pages in page_0 format
	01h to FEh	Reserved
	FFh	Return all subpages for all mode pages in the page_0 format for subpage 00h and in the sub_page format for subpages 01h to FEh

The ALLOCATION LENGTH field is defined in 4.3.5.6.

An application client may request any one or all of the supported mode pages from the device server. If an application client issues a MODE SENSE command with a page code or subpage code value not implemented by the logical unit, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

If an application client requests all supported mode pages, the device server shall return the supported pages in ascending page code order beginning with mode page 01h. If mode page 00h is implemented, the device server shall return mode page 00h after all other mode pages have been returned.

If the PC field and the PAGE CODE field are both set to zero, the device server should return a mode parameter header and block descriptor, if applicable.

The mode parameter list for all device types for MODE SELECT and MODE SENSE is defined in 7.4. Parts of the mode parameter list are specifically defined for each device type. Definitions for the parts of each mode parameter list that are unique for each device-type may be found in the applicable command standards (see 3.1.27).

#### **6.11.2 Current values**

A PC field value of 00b requests that the device server return the current values of the mode parameters. The current values returned are:

- a) The current values of the mode parameters established by the last successful MODE SELECT command;
- b) The saved values of the mode parameters if a MODE SELECT command has not successfully completed since the mode parameters were restored to their saved values (see 6.9); or
- c) The default values of the mode parameters if a MODE SELECT command has not successfully completed since the mode parameters were restored to their default values (see 6.9).

#### **6.11.3 Changeable values**

A PC field value of 01b requests that the device server return a mask denoting those mode parameters that are changeable. In the mask, the bits in the fields of the mode parameters that are changeable all shall be set to one and the bits in the fields of the mode parameters that are non-changeable (i.e., defined by the logical unit) all shall be set to zero.

If the logical unit does not implement changeable parameters mode pages and the device server receives a MODE SENSE command with 01b in the PC field, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

An attempt to change a non-changeable mode parameter using the MODE SELECT command shall result in an error condition (see 6.9).

The application client should issue a MODE SENSE command with the PC field set to 01b and the PAGE CODE field set to 3Fh to determine which mode pages are supported, which mode parameters within the mode pages are changeable, and the supported length of each mode page prior to issuing any MODE SELECT commands.

#### **6.11.4 Default values**

A PC field value of 10b requests that the device server return the default values of the mode parameters. Unsupported parameters shall be set to zero. Default values should be accessible even if the logical unit is not ready.

#### **6.11.5 Saved values**

A PC field value of 11b requests that the device server return the saved values of the mode parameters. Mode parameters not supported by the logical unit shall be set to zero. If saved values are not implemented, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to SAVING PARAMETERS NOT SUPPORTED.

The method of saving parameters is vendor specific. The parameters are preserved in such a manner that they are retained when the device is powered down. All saveable mode pages should be considered saved when a MODE



SELECT command issued with the SP bit set to one has completed with a GOOD status or after the successful completion of a FORMAT UNIT command.

### 6.11.6 Initial responses

After a logical unit reset, the device server shall respond in the following manner:

- a) If default values are requested, report the default values;
- b) If saved values are requested, report valid restored mode parameters, or restore the mode parameters and report them. If the saved values of the mode parameters are not able to be accessed from the nonvolatile vendor specific location, the command shall be terminated with CHECK CONDITION status, with the sense key set to NOT READY. If saved parameters are not implemented, respond as defined in 6.11.5; or
- c) If current values are requested and the current values have been sent by the application client via a MODE SELECT command, the current values shall be returned. If the current values have not been sent, the device server shall return:
  - A) The saved values, if saving is implemented and saved values are available; or
  - B) The default values.

## 6.12 MODE SENSE(10) command

The MODE SENSE(10) command (see table 156) provides a means for a device server to report parameters to an application client. It is a complementary command to the MODE SELECT(10) command. Device servers that implement the MODE SENSE(10) command shall also implement the MODE SELECT(10) command.

**Table 156 — MODE SENSE(10) command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (5Ah)							
1	Reserved			LLBAA	DBD	Reserved		
2	PC		PAGE CODE					
3	SUBPAGE CODE							
4	Reserved							
6								
7	(MSB)							
8	ALLOCATION LENGTH							
9	(LSB)							
	CONTROL							

If the Long LBA Accepted (LLBAA) bit is set to one, the device server is allowed to return parameter data with the LONGLBA bit equal to one (see 7.4.3). If LLBAA bit is set to zero, the LONGLBA bit shall be zero in the parameter data returned by the device server.

See the MODE SENSE(6) command (6.11) for a description of the other fields and operation of this command.

## 6.13 PERSISTENT RESERVE IN command

### 6.13.1 PERSISTENT RESERVE IN command introduction

The PERSISTENT RESERVE IN command (see table 157) is used to obtain information about persistent reservations and reservation keys (i.e., registrations) that are active within a device server. This command is used in conjunction with the PERSISTENT RESERVE OUT command (see 6.14).

**Table 157 — PERSISTENT RESERVE IN command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (5Eh)							
1	Reserved			SERVICE ACTION				
2	Reserved							
6								
7	(MSB)	ALLOCATION LENGTH						(LSB)
8								
9	CONTROL							

The ALLOCATION LENGTH field is defined in 4.3.5.6. The PERSISTENT RESERVE IN parameter data includes a length field that indicates the number of parameter data bytes available to be returned. The allocation length should be set to a value large enough to return the length field for the specified service action.

The service action codes for the PERSISTENT RESERVE IN command are defined in table 158.

**Table 158 — PERSISTENT RESERVE IN service action codes**

Code	Name	Description	Reference
00h	READ KEYS	Reads all registered reservation keys (i.e., registrations) as described in 5.7.6.2	6.13.2
01h	READ RESERVATION	Reads the current persistent reservations as described in 5.7.6.3	6.13.3
02h	REPORT CAPABILITIES	Returns capability information	6.13.4
03h	READ FULL STATUS	Reads complete information about all registrations and the persistent reservations, if any	6.13.5
04h to 1Fh	Reserved	Reserved	

### 6.13.2 READ KEYS service action

The READ KEYS service action requests that the device server return a parameter list containing a header and a list of each currently registered I\_T nexus' reservation key. If multiple I\_T nexuses have registered with the same key, then that key value shall be listed multiple times, once for each such registration.

For more information on READ KEYS see 5.7.6.2.

The format for the parameter data provided in response to a PERSISTENT RESERVE IN command with the READ KEYS service action is shown in table 159.

**Table 159 — PERSISTENT RESERVE IN parameter data for READ KEYS**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)	PRGENERATION						(LSB)
3								
4	(MSB)	ADDITIONAL LENGTH (n-7)						(LSB)
7								
		Reservation key list						
8	(MSB)	Reservation key [first]						(LSB)
15								
		⋮						
n-7	(MSB)	Reservation key [last]						(LSB)
n								

The Persistent Reservations Generation (PRGENERATION) field shall contain the value of a 32-bit wrapping counter maintained by the device server that shall be incremented every time a PERSISTENT RESERVE OUT command requests a REGISTER service action, a REGISTER AND IGNORE EXISTING KEY service action, a REGISTER AND MOVE service action, a CLEAR service action, a PREEMPT service action, or a PREEMPT AND ABORT service action. The counter shall not be incremented by a PERSISTENT RESERVE IN command, by a PERSISTENT RESERVE OUT command that performs a RESERVE or RELEASE service action, or by a PERSISTENT RESERVE OUT command that is terminated due to an error or reservation conflict. Regardless of the APTPL bit value the PRgeneration value shall be set to zero by a power on.

The ADDITIONAL LENGTH field contains a count of the number of bytes in the Reservation key list. The relationship between the ADDITIONAL LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The reservation key list contains the 8-byte reservation keys for all I\_T nexuses that have been registered (see 5.7.7).

### 6.13.3 READ RESERVATION service action

#### 6.13.3.1 READ RESERVATION service action introduction

The READ RESERVATION service action requests that the device server return a parameter list containing a header and the persistent reservation, if any, that is present in the device server.

For more information on READ RESERVATION see 5.7.6.3.

### 6.13.3.2 Format of PERSISTENT RESERVE IN parameter data for READ RESERVATION

When no persistent reservation is held, the format for the parameter data provided in response to a PERSISTENT RESERVE IN command with the READ RESERVATION service action is shown in table 160.

**Table 160 — PERSISTENT RESERVE IN parameter data for READ RESERVATION with no reservation held**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____							
3	PRGENERATION (LSB)							
4	(MSB) _____							
7	ADDITIONAL LENGTH (0) (LSB)							

The PRGENERATION field shall be as defined for the PERSISTENT RESERVE IN command with READ KEYS service action parameter data (see 6.13.2).

The ADDITIONAL LENGTH field shall be set to zero, indicating that no persistent reservation is held.

When a persistent reservation is held, the format for the parameter data provided in response to a PERSISTENT RESERVE IN command with the READ RESERVATION service action is shown in table 161.

**Table 161 — PERSISTENT RESERVE IN parameter data for READ RESERVATION with reservation**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____							
3	PRGENERATION (LSB)							
4	(MSB) _____							
7	ADDITIONAL LENGTH (10h) (LSB)							
8	(MSB) _____							
15	RESERVATION KEY (LSB)							
16	Obsolete							
19								
20	Reserved							
21	SCOPE				TYPE			
22	Obsolete							
23								

The PRGENERATION field shall be as defined for the PERSISTENT RESERVE IN command with READ KEYS service action parameter data.

The ADDITIONAL LENGTH field contains a count of the number of bytes to follow and shall be set to 16. The relationship between the ADDITIONAL LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The RESERVATION KEY field shall contain the reservation key under which the persistent reservation is held (see 5.7.10).

The SCOPE field shall be set to LU\_SCOPE (see 6.13.3.3).

The TYPE field shall contain the persistent reservation type (see 6.13.3.4) specified in the PERSISTENT RESERVE OUT command that created the persistent reservation.

The obsolete fields in bytes 16 to 19, byte 22, and byte 23 were defined in a previous standard.

### 6.13.3.3 Persistent reservations scope

The SCOPE field (see table 162) shall be set to LU\_SCOPE, specifying that the persistent reservation applies to the entire logical unit.

**Table 162 — Persistent reservation SCOPE field**

Code	Name	Description
0h	LU_SCOPE	Persistent reservation applies to the full logical unit
1h to 2h		Obsolete
3h to Fh		Reserved

The LU\_SCOPE scope shall be implemented by all device servers that implement PERSISTENT RESERVE OUT.

### 6.13.3.4 Persistent reservations type

The TYPE field (see table 163) specifies the characteristics of the persistent reservation being established for all logical blocks within the logical unit. Table 44 (see 5.7.1) defines the persistent reservation types under which each command defined in this standard is allowed to be processed. Each other command standard (see 3.1.27) defines the persistent reservation types under which each command defined in that command standard is allowed to be processed.

**Table 163 — Persistent reservation TYPE field (part 1 of 2)**

Code	Name	Description
0h		Obsolete
1h	Write Exclusive	<b>Access Restrictions:</b> Some commands (e.g., media-access write commands) are only allowed for the persistent reservation holder (see 5.7.10). <b>Persistent Reservation Holder:</b> There is only one persistent reservation holder.
2h		Obsolete
3h	Exclusive Access	<b>Access Restrictions:</b> Some commands (e.g., media-access commands) are only allowed for the persistent reservation holder (see 5.7.10). <b>Persistent Reservation Holder:</b> There is only one persistent reservation holder.
4h		Obsolete

**Table 163 — Persistent reservation TYPE field (part 2 of 2)**

Code	Name	Description
5h	Write Exclusive – Registrants Only	<b>Access Restrictions:</b> Some commands (e.g., media-access write commands) are only allowed for registered I_T nexuses. <b>Persistent Reservation Holder:</b> There is only one persistent reservation holder (see 5.7.10).
6h	Exclusive Access – Registrants Only	<b>Access Restrictions:</b> Some commands (e.g., media-access commands) are only allowed for registered I_T nexuses. <b>Persistent Reservation Holder:</b> There is only one persistent reservation holder (see 5.7.10).
7h	Write Exclusive – All Registrants	<b>Access Restrictions:</b> Some commands (e.g., media-access write commands) are only allowed for registered I_T nexuses. <b>Persistent Reservation Holder:</b> Each registered I_T nexus is a persistent reservation holder (see 5.7.10).
8h	Exclusive Access – All Registrants	<b>Access Restrictions:</b> Some commands (e.g., media-access commands) are only allowed for registered I_T nexuses. <b>Persistent Reservation Holder:</b> Each registered I_T nexus is a persistent reservation holder (see 5.7.10).
9h to Fh	Reserved	

**6.13.4 REPORT CAPABILITIES service action**

The REPORT CAPABILITIES service action requests that the device server return information on persistent reservation features.

The format for the parameter data provided in response to a PERSISTENT RESERVE IN command with the REPORT CAPABILITIES service action is shown in table 164.

**Table 164 — PERSISTENT RESERVE IN parameter data for REPORT CAPABILITIES**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____							
1	LENGTH (0008h) _____ (LSB)							
2	Reserved			CRH	SIP_C	ATP_C	Reserved	PTPL_C
3	TMV	ALLOW COMMANDS			Reserved			PTPL_A
4	_____							
5	PERSISTENT RESERVATION TYPE MASK _____							
6	_____							
7	Reserved _____							

The LENGTH field indicates the length in bytes of the parameter data. The relationship between the LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

A Compatible Reservation Handling (CRH) bit set to one indicates that the device server supports the exceptions to the SPC-2 RESERVE and RELEASE commands described in 5.7.3. A CRH bit set to zero indicates that

RESERVE(6) command, RESERVE(10) command, RELEASE(6) command, and RELEASE(10) command are processed as defined in SPC-2.

A Specify Initiator Ports Capable (SIP\_C) bit set to one indicates that the device server supports the SPEC\_I\_PT bit in the PERSISTENT RESERVE OUT command parameter data (see 6.14.3). An SIP\_C bit set to zero indicates that the device server does not support the SPEC\_I\_PT bit in the PERSISTENT RESERVE OUT command parameter data.

An All Target Ports Capable (ATP\_C) bit set to one indicates that the device server supports the ALL\_TG\_PT bit in the PERSISTENT RESERVE OUT command parameter data. An ATP\_C bit set to zero indicates that the device server does not support the ALL\_TG\_PT bit in the PERSISTENT RESERVE OUT command parameter data.

A Persist Through Power Loss Capable (PTPL\_C) bit set to one indicates that the device server supports the persist through power loss capability (see 5.7.5) for persistent reservations and the APTPL bit in the PERSISTENT RESERVE OUT command parameter data. An PTPL\_C bit set to zero indicates that the device server does not support the persist through power loss capability.

A Type Mask Valid (TMV) bit set to one indicates that the PERSISTENT RESERVATION TYPE MASK field contains a bit map indicating which persistent reservation types are supported by the device server. A TMV bit set to zero indicates that the PERSISTENT RESERVATION TYPE MASK field shall be ignored.

The ALLOW COMMANDS field (see table 165) indicates whether certain commands are allowed through certain types of persistent reservations.

**Table 165 — ALLOW COMMANDS field**

Code	Description
000b	No information is provided about whether certain commands are allowed through certain types of persistent reservations.
001b	The device server allows the TEST UNIT READY command (see table 44 in 5.7.1) through Write Exclusive and Exclusive Access persistent reservations and does not provide information about whether the following commands are allowed through Write Exclusive persistent reservations: <ul style="list-style-type: none"> <li>a) the MODE SENSE, READ ATTRIBUTE, READ BUFFER, RECEIVE DIAGNOSTIC RESULTS, REPORT SUPPORTED OPERATION CODES, and REPORT SUPPORTED TASK MANAGEMENT FUNCTION commands (see table 44 in 5.7.1); and</li> <li>b) the READ DEFECT DATA command (see SBC-3).</li> </ul>
010b	The device server allows the TEST UNIT READY command through Write Exclusive and Exclusive Access persistent reservations and does not allow the following commands through Write Exclusive persistent reservations: <ul style="list-style-type: none"> <li>a) the MODE SENSE, READ ATTRIBUTE, READ BUFFER, RECEIVE DIAGNOSTIC RESULTS, REPORT SUPPORTED OPERATION CODES, and REPORT SUPPORTED TASK MANAGEMENT FUNCTION commands; and</li> <li>b) the READ DEFECT DATA command.</li> </ul>
011b	The device server allows the TEST UNIT READY command through Write Exclusive and Exclusive Access persistent reservations and allows the following commands through Write Exclusive persistent reservations: <ul style="list-style-type: none"> <li>a) the MODE SENSE, READ ATTRIBUTE, READ BUFFER, RECEIVE DIAGNOSTIC RESULTS, REPORT SUPPORTED OPERATION CODES, and REPORT SUPPORTED TASK MANAGEMENT FUNCTION commands; and</li> <li>b) the READ DEFECT DATA command.</li> </ul>
100b to 111b	Reserved

A Persist Through Power Loss Activated (PTPL\_A) bit set to one indicates that the persist through power loss capability is activated (see 5.7.5). A PTPL\_A bit set to zero indicates that the persist through power loss capability is not activated.

The PERSISTENT RESERVATION TYPE MASK field (see table 166) contains a bit map that indicates the persistent reservation types that are supported by the device server.

**Table 166 — Persistent Reservation Type Mask format**

Bit Byte	7	6	5	4	3	2	1	0
4	WR_EX_AR	EX_AC_RO	WR_EX_RO	Reserved	EX_AC	Reserved	WR_EX	Reserved
5	Reserved							EX_AC_AR

A Write Exclusive – All Registrants (WR\_EX\_AR) bit set to one indicates that the device server supports the Write Exclusive – All Registrants persistent reservation type. An WR\_EX\_AR bit set to zero indicates that the device server does not support the Write Exclusive – All Registrants persistent reservation type.

An Exclusive Access – Registrants Only (EX\_AC\_RO) bit set to one indicates that the device server supports the Exclusive Access – Registrants Only persistent reservation type. An EX\_AC\_RO bit set to zero indicates that the device server does not support the Exclusive Access – Registrants Only persistent reservation type.

A Write Exclusive – Registrants Only (WR\_EX\_RO) bit set to one indicates that the device server supports the Write Exclusive – Registrants Only persistent reservation type. An WR\_EX\_RO bit set to zero indicates that the device server does not support the Write Exclusive – Registrants Only persistent reservation type.

An Exclusive Access (EX\_AC) bit set to one indicates that the device server supports the Exclusive Access persistent reservation type. An EX\_AC bit set to zero indicates that the device server does not support the Exclusive Access persistent reservation type.

A Write Exclusive (WR\_EX) bit set to one indicates that the device server supports the Write Exclusive persistent reservation type. An WR\_EX bit set to zero indicates that the device server does not support the Write Exclusive persistent reservation type.

An Exclusive Access – All Registrants (EX\_AC\_AR) bit set to one indicates that the device server supports the Exclusive Access – All Registrants persistent reservation type. An EX\_AC\_AR bit set to zero indicates that the device server does not support the Exclusive Access – All Registrants persistent reservation type.



### 6.13.5 READ FULL STATUS service action

The READ FULL STATUS service action requests that the device server return a parameter list describing the registration and persistent reservation status of each currently registered I\_T nexus for the logical unit.

For more information on READ FULL STATUS see 5.7.6.4.

The format for the parameter data provided in response to a PERSISTENT RESERVE IN command with the READ FULL STATUS service action is shown in table 167.

**Table 167 — PERSISTENT RESERVE IN parameter data for READ FULL STATUS**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____ PRGENERATION _____ (LSB)							
3								
4	(MSB) _____ ADDITIONAL LENGTH (n-7) _____ (LSB)							
7								
	Full status descriptors							
8	Full status descriptor [first] (see table 168) _____							
	⋮							
n	Full status descriptor [last] (see table 168) _____							

The PRGENERATION field shall be as defined for the PERSISTENT RESERVE IN command with READ KEYS service action parameter data (see 6.13.2).

The ADDITIONAL LENGTH field contains a count of the number of bytes to follow in the full status descriptors. The relationship between the ADDITIONAL LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The format of the full status descriptors is shown in table 168. Each full status descriptor describes one or more registered I\_T nexuses. The device server shall return persistent reservations status information for every registered I\_T nexus.

**Table 168 — PERSISTENT RESERVE IN full status descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
7	RESERVATION KEY							(LSB)
8	Reserved							
11								
12	Reserved						ALL_TG_PT	R HOLDER
13	SCOPE				TYPE			
14	Reserved							
17								
18	(MSB)							
19	RELATIVE TARGET PORT IDENTIFIER							(LSB)
20	(MSB)							
23	ADDITIONAL DESCRIPTOR LENGTH (n-23)							(LSB)
24	TRANSPORTID							
n								

The RESERVATION KEY field contains the reservation key.

A Reservation Holder (R HOLDER) bit set to one indicates that all I\_T nexuses described by this full status descriptor are registered and are persistent reservation holders (see 5.7.10). A R HOLDER bit set to zero indicates that all I\_T nexuses described by this full status descriptor are registered but are not persistent reservation holders.

An All Target Ports (ALL\_TG\_PT) bit set to zero indicates that this full status descriptor represents a single I\_T nexus. An ALL\_TG\_PT bit set to one indicates that:

- a) This full status descriptor represents all the I\_T nexuses that are associated with both:
  - A) The initiator port specified by the TRANSPORTID field; and
  - B) Every target port in the SCSI target device;
- b) All the I\_T nexuses are registered with the same reservation key; and
- c) All the I\_T nexuses are either reservation holders or not reservation holders as indicated by the R HOLDER bit.

The device server is not required to return an ALL\_TG\_PT bit set to one. Instead, it may return separate full status descriptors for each I\_T nexus.

If the R HOLDER bit is set to one (i.e., if the I\_T nexus described by this full status descriptor is a reservation holder), the SCOPE field and the TYPE field are as defined in the READ RESERVATION service action parameter data (see 6.13.3). If the R HOLDER bit is set to zero, the contents of the SCOPE field and the TYPE field are not defined by this standard.

If the ALL\_TG\_PT bit set to zero, the RELATIVE TARGET PORT IDENTIFIER field contains the relative port identifier (see 3.1.120) of the target port that is part of the I\_T nexus described by this full status descriptor. If the ALL\_TG\_PT bit is set to one, the contents of the RELATIVE TARGET PORT IDENTIFIER field are not defined by this standard.

The ADDITIONAL DESCRIPTOR LENGTH field contains a count of the number of bytes that follow in the descriptor (i.e., the size of the TransportID).

The TRANSPORTID field contains a TransportID (see 7.5.4) identifying the initiator port that is part of the I\_T nexus or I\_T nexuses described by this full status descriptor.

6.14 PERSISTENT RESERVE OUT command

6.14.1 PERSISTENT RESERVE OUT command introduction

The PERSISTENT RESERVE OUT command (see table 169) is used to request service actions that reserve a logical unit for the exclusive or shared use of a particular I\_T nexus. The command uses other service actions to manage and remove such persistent reservations.

I\_T nexuses performing PERSISTENT RESERVE OUT service actions are identified by a registered reservation key provided by the application client. An application client may use the PERSISTENT RESERVE IN command to obtain the reservation key, if any, for the I\_T nexus holding a persistent reservation and may use the PERSISTENT RESERVE OUT command to preempt that persistent reservation.

Table 169 — PERSISTENT RESERVE OUT command

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (5Fh)							
1	Reserved			SERVICE ACTION				
2	SCOPE				TYPE			
3	Reserved							
4								
5	PARAMETER LIST LENGTH							
8								
9	CONTROL							

If a PERSISTENT RESERVE OUT command is attempted, but there are insufficient device server resources to complete the operation, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INSUFFICIENT REGISTRATION RESOURCES.

The PERSISTENT RESERVE OUT command contains fields that specify a persistent reservation service action, the intended scope of the persistent reservation, and the restrictions caused by the persistent reservation. The SCOPE field and TYPE field are defined in 6.13.3.3 and 6.13.3.4. If a SCOPE field specifies a scope that is not implemented, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

Fields contained in the PERSISTENT RESERVE OUT parameter list specify the information required to perform a particular persistent reservation service action.

The PARAMETER LIST LENGTH field specifies the number of bytes of parameter data for the PERSISTENT RESERVE OUT command.

The parameter list shall be 24 bytes in length and the PARAMETER LIST LENGTH field shall contain 24 (18h), if the following conditions are true:

- a) The SPEC\_I\_PT bit (see 6.14.3) is set to zero; and
- b) The service action is not REGISTER AND MOVE.

If the SPEC\_I\_PT bit is set to zero, the service action is not REGISTER AND MOVE, and the parameter list length is not 24, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

If the parameter list length is larger than the device server is able to process, the command should be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

### 6.14.2 PERSISTENT RESERVE OUT service actions

When processing the PERSISTENT RESERVE OUT service actions, the device server shall increment the PRgen-eration value as specified in 6.13.2.

The PERSISTENT RESERVE OUT command service actions are defined in table 170.

**Table 170 — PERSISTENT RESERVE OUT service action codes**

Code	Name	Description	PRGENERATION field incremented (see 6.13.2)	Parameter list format
00h	REGISTER	Register a reservation key with the device server (see 5.7.7) or unregister a reservation key (see 5.7.11.3).	Yes	Basic (see 6.14.3)
01h	RESERVE	Creates a persistent reservation having a specified SCOPE and TYPE (see 5.7.9). The SCOPE and TYPE of a persistent reservation are defined in 6.13.3.3 and 6.13.3.4.	No	Basic (see 6.14.3)
02h	RELEASE	Releases the selected persistent reservation (see 5.7.11.2).	No	Basic (see 6.14.3)
03h	CLEAR	Clears all reservation keys (i.e., registrations) and all persistent reservations (see 5.7.11.6).	Yes	Basic (see 6.14.3)
04h	PREEMPT	Preempts persistent reservations and/or removes registrations (see 5.7.11.4).	Yes	Basic (see 6.14.3)
05h	PREEMPT AND ABORT	Preempts persistent reservations and/or removes registrations and aborts all tasks for all preempted I_T nexuses (see 5.7.11.4 and 5.7.11.5).	Yes	Basic (see 6.14.3)
06h	REGISTER AND IGNORE EXISTING KEY	Register a reservation key with the device server (see 5.7.7) or unregister a reservation key (see 5.7.11.3).	Yes	Basic (see 6.14.3)
07h	REGISTER AND MOVE	Register a reservation key for another I_T nexus with the device server and move a persistent reservation to that I_T nexus (see 5.7.8)	Yes	Register and move (see 6.14.4)
08h to 1Fh	Reserved			

### 6.14.3 Basic PERSISTENT RESERVE OUT parameter list

The parameter list format shown in table 171 shall be used by the PERSISTENT RESERVE OUT command with any service action except the REGISTER AND MOVE service action. All fields shall be sent, even if the field is not required for the specified service action and scope values.

**Table 171 — PERSISTENT RESERVE OUT parameter list**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
7	RESERVATION KEY (LSB)							
8	(MSB)							
15	SERVICE ACTION RESERVATION KEY (LSB)							
16	Obsolete							
19								
20	Reserved				SPEC_I_PT	ALL_TG_PT	Reserved	APTPL
21	Reserved							
22	Obsolete							
23								
24	Additional parameter data							
n								

The obsolete fields in bytes 16 to 19, byte 22 and byte 23 were defined in a previous standard.

The RESERVATION KEY field contains an 8-byte value provided by the application client to the device server to identify the I\_T nexus that is the source of the PERSISTENT RESERVE OUT command. The device server shall verify that the contents of the RESERVATION KEY field in a PERSISTENT RESERVE OUT command parameter data matches the registered reservation key for the I\_T nexus from which the command was received, except for:

- a) The REGISTER AND IGNORE EXISTING KEY service action where the RESERVATION KEY field shall be ignored; and
- b) The REGISTER service action for an unregistered I\_T nexus where the RESERVATION KEY field shall contain zero.

Except as noted above, when a PERSISTENT RESERVE OUT command specifies a RESERVATION KEY field other than the reservation key registered for the I\_T nexus the device server shall complete the command with RESERVATION CONFLICT status. Except as noted above, the reservation key of the I\_T nexus shall be verified to be correct regardless of the SERVICE ACTION and SCOPE field values.

The SERVICE ACTION RESERVATION KEY field contains information needed for the following service actions: REGISTER, REGISTER AND IGNORE EXISTING KEY, PREEMPT, and PREEMPT AND ABORT. The SERVICE ACTION RESERVATION KEY field is ignored for the following service actions: RESERVE, RELEASE, and CLEAR.

For the REGISTER service action and REGISTER AND IGNORE EXISTING KEY service action, the SERVICE ACTION RESERVATION KEY field contains:

- a) The new reservation key to be registered in place of the registered reservation key; or
- b) Zero to unregister the registered reservation key.

For the PREEMPT service action and PREEMPT AND ABORT service action, the SERVICE ACTION RESERVATION KEY field contains the reservation key of:

- a) The registrations to be removed; and
- b) If the SERVICE ACTION RESERVATION KEY field identifies a persistent reservation holder (see 5.7.10), persistent reservations that are to be preempted.

If the Specify Initiator Ports (SPEC\_I\_PT) bit is set to zero, the device server shall apply the registration only to the I\_T nexus that sent the PERSISTENT RESERVE OUT command. If the SPEC\_I\_PT bit is set to one for any service action except the REGISTER service action, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST. If the SPEC\_I\_PT bit is set to one for the REGISTER service action, the additional parameter data (see table 172) shall include a list of transport IDs and the device server shall also apply the registration to the I\_T nexus for each initiator port specified by a TransportID. If a registration fails for any initiator port (e.g., if the logical unit does not have enough resources available to hold the registration information), no registrations shall be made, and the command shall be terminated with CHECK CONDITION status.

**Table 172 — PERSISTENT RESERVE OUT specify initiator ports additional parameter data**

Bit Byte	7	6	5	4	3	2	1	0
24	TRANSPORTID PARAMETER DATA LENGTH (n-27)							
27								
	TransportIDs list							
28	TransportID [first]							
	⋮							
	TransportID [last]							
n								

The TRANSPORTID PARAMETER DATA LENGTH field specifies the number of bytes of TransportIDs that follow.

The command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST:

- a) If the value in the PARAMETER LIST LENGTH field in the CDB does not include all of the additional parameter list bytes specified by the TRANSPORTID PARAMETER DATA LENGTH field; or
- b) If the value in the TRANSPORTID PARAMETER DATA LENGTH field results in the truncation of a TransportID.

The format of a TransportID is specified in 7.5.4.

The All Target Ports (ALL\_TG\_PT) bit is valid only for the REGISTER service action and the REGISTER AND IGNORE EXISTING KEY service action, and shall be ignored for all other service actions. Support for the ALL\_TG\_PT bit is optional. If the device server receives a REGISTER service action or a REGISTER AND IGNORE EXISTING KEY service action with the ALL\_TG\_PT bit set to one, it shall create the specified registration on all target ports in the SCSI target device known to the device server (i.e., as if the same registration request had been received individually through each target port). If the device server receives a REGISTER service action or a REGISTER AND IGNORE EXISTING KEY service action with the ALL\_TG\_PT bit set to zero, it shall apply the registration only to the target port through which the PERSISTENT RESERVE OUT command was received. If a device

server that does not support an ALL\_TG\_PT bit set to one receives that value in a REGISTER service action or a REGISTER AND IGNORE EXISTING KEY service action, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The Activate Persist Through Power Loss (APTPL) bit is valid only for the REGISTER service action and the REGISTER AND IGNORE EXISTING KEY service action, and shall be ignored for all other service actions. Support for an APTPL bit equal to one is optional. If a device server that does not support an APTPL bit set to one receives that value in a REGISTER service action or a REGISTER AND IGNORE EXISTING KEY service action, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the last valid APTPL bit value received by the device server is zero, the loss of power in the SCSI target device shall release the persistent reservation for the logical unit and remove all registered reservation keys (see 5.7.7). If the last valid APTPL bit value received by the device server is one, the logical unit shall retain any persistent reservation(s) that may be present and all reservation keys (i.e., registrations) for all I\_T nexuses even if power is lost and later returned (see 5.7.5).

Table 173 summarizes which fields are set by the application client and interpreted by the device server for each service action and scope value.

**Table 173 — PERSISTENT RESERVE OUT service actions and valid parameters (part 1 of 2)**

Service action	Allowed SCOPE	Parameters (part 1 of 2)			
		TYPE	RESERVATION KEY	SERVICE ACTION RESERVATION KEY	APTPL
REGISTER	ignored	ignored	valid	valid	valid
REGISTER AND IGNORE EXISTING KEY	ignored	ignored	ignored	valid	valid
RESERVE	LU_SCOPE	valid	valid	ignored	ignored
RELEASE	LU_SCOPE	valid	valid	ignored	ignored
CLEAR	ignored	ignored	valid	ignored	ignored
PREEMPT	LU_SCOPE	valid	valid	valid	ignored
PREEMPT AND ABORT	LU_SCOPE	valid	valid	valid	ignored
REGISTER AND MOVE	LU_SCOPE	valid	valid	valid	not applicable <sup>a</sup>
<sup>a</sup> The parameter list format for the REGISTER AND MOVE service action is described in 6.14.4.					



**Table 173 — PERSISTENT RESERVE OUT service actions and valid parameters (part 2 of 2)**

Service action	Allowed SCOPE	Parameters (part 2 of 2)	
		ALL_TG_PT	SPEC_I_PT
REGISTER	ignored	valid	valid
REGISTER AND IGNORE EXISTING KEY	ignored	valid	invalid
RESERVE	LU_SCOPE	ignored	invalid
RELEASE	LU_SCOPE	ignored	invalid
CLEAR	ignored	ignored	invalid
PREEMPT	LU_SCOPE	ignored	invalid
PREEMPT AND ABORT	LU_SCOPE	ignored	invalid
REGISTER AND MOVE	LU_SCOPE	not applicable <sup>a</sup>	invalid
<sup>a</sup> The parameter list format for the REGISTER AND MOVE service action is described in 6.14.4.			

**6.14.4 PERSISTENT RESERVE OUT command with REGISTER AND MOVE service action parameters**

The parameter list format shown in table 174 shall be used by the PERSISTENT RESERVE OUT command with REGISTER AND MOVE service action.

**Table 174 — PERSISTENT RESERVE OUT command with REGISTER AND MOVE service action parameter list**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)	RESERVATION KEY						(LSB)
7								
8	(MSB)	SERVICE ACTION RESERVATION KEY						(LSB)
15								
16		Reserved						
17		Reserved					UNREG	APTPL
18	(MSB)	RELATIVE TARGET PORT IDENTIFIER						(LSB)
19								
20	(MSB)	TRANSPORTID PARAMETER DATA LENGTH (n-23)						(LSB)
23								
24		TransportID						
n								

The RESERVATION KEY field contains an 8-byte value provided by the application client to the device server to identify the I\_T nexus that is the source of the PERSISTENT RESERVE OUT command. The device server shall verify that the contents of the RESERVATION KEY field in a PERSISTENT RESERVE OUT command parameter data matches the registered reservation key for the I\_T nexus from which the command was received. If a PERSISTENT RESERVE OUT command specifies a RESERVATION KEY field other than the reservation key registered for the I\_T nexus, the device server shall complete the command with RESERVATION CONFLICT status.

The SERVICE ACTION RESERVATION KEY field contains the reservation key to be registered to the specified I\_T nexus.

The Activate Persist Through Power Loss (APTPL) bit set to one is optional. If a device server that does not support an APTPL bit set to one receives that value, it shall return CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the last valid APTPL bit value received by the device server is zero, the loss of power in the SCSI target device shall release the persistent reservation for the logical unit and remove all registered reservation keys (see 5.6.5). If the last valid APTPL bit value received by the device server is one, the logical unit shall retain any persistent reservation(s) that may be present and all reservation keys (i.e., registrations) for all I\_T nexuses even if power is lost and later returned (see 5.7.5).

The unregister (UNREG) bit set to zero specifies that the device server shall not unregister the I\_T nexus on which the PERSISTENT RESERVE OUT command REGISTER AND MOVE service action was received. An UNREG bit set to one specifies that the device server shall unregister the I\_T nexus on which the PERSISTENT RESERVE OUT command REGISTER AND MOVE service action was received.

The RELATIVE TARGET PORT IDENTIFIER field specifies the relative port identifier (see 3.1.120) of the target port in the I\_T nexus to which the persistent reservation is to be moved.

The TRANSPORTID DESCRIPTOR LENGTH field specifies the number of bytes of the TransportID that follows, shall be a minimum of 24 bytes, and shall be a multiple of 4.

The TransportID specifies the initiator port in the I\_T nexus to which the persistent reservation is to be moved. The format of the TransportID is defined in 7.5.4.

The command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST:

- a) If the value in the PARAMETER LIST LENGTH field in the CDB does not include all of the parameter list bytes specified by the TRANSPORTID PARAMETER DATA LENGTH field; or
- b) If the value in the TRANSPORTID PARAMETER DATA LENGTH field results in the truncation of a TransportID.

## 6.15 READ ATTRIBUTE command

### 6.15.1 READ ATTRIBUTE command introduction

The READ ATTRIBUTE command (see table 175) allows an application client to read attribute values from medium auxiliary memory.

**Table 175 — READ ATTRIBUTE command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (8Ch)							
1	Reserved			SERVICE ACTION				
2	Restricted (see SMC-2)							
4								
5	VOLUME NUMBER							
6	Reserved							
7	PARTITION NUMBER							
8	(MSB)	FIRST ATTRIBUTE IDENTIFIER						
9								(LSB)
10	(MSB)	ALLOCATION LENGTH						
13								(LSB)
14	Reserved							
15	CONTROL							

If the medium auxiliary memory is not accessible because there is no medium present, the READ ATTRIBUTE command shall be terminated with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to MEDIUM NOT PRESENT.

If the medium is present but the medium auxiliary memory is not accessible, the READ ATTRIBUTE command shall be terminated with CHECK CONDITION status, with the sense key set to MEDIUM ERROR, and the additional sense code set to LOGICAL UNIT NOT READY, AUXILIARY MEMORY NOT ACCESSIBLE.

If the medium auxiliary memory is not operational, the READ ATTRIBUTE command shall be terminated with CHECK CONDITION status, with the sense key set to MEDIUM ERROR, and the additional sense code set to AUXILIARY MEMORY READ ERROR.

The service actions defined for the READ ATTRIBUTE command are shown in table 176.

**Table 176 — READ ATTRIBUTE service action codes**

Code	Name	Description	Reference
00h	ATTRIBUTE VALUES	Return attribute values.	6.15.2
01h	ATTRIBUTE LIST	Return a list of available attribute identifiers, identifiers that are not in the nonexistent state or unsupported state (see 5.11).	6.15.3
02h	VOLUME LIST	Return a list of known volume numbers.	6.15.4
03h	PARTITION LIST	Return a list of known partition numbers.	6.15.5
04h	Restricted		
05h to 1Fh	Reserved		

The VOLUME NUMBER field specifies a volume (see SSC-2) within the medium auxiliary memory. The number of volumes of the medium auxiliary memory shall equal that of the attached medium. If the medium only has a single volume, then its volume number shall be zero.

The PARTITION NUMBER field specifies a partition (see SSC-2) within a volume. The number of partitions of the medium auxiliary memory shall equal that of the attached medium. If the medium only has a single partition, then its partition number shall be zero.

If the combination of volume number and partition number is not valid, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The FIRST ATTRIBUTE IDENTIFIER field specifies the attribute identifier of the first attribute to be returned. If the specified attribute is in the unsupported state or nonexistent state (see 5.11), the READ ATTRIBUTE command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The ALLOCATION LENGTH field is defined in 4.3.5.6.

The format of parameter data returned by the READ ATTRIBUTE command depends on the service action specified.

### **6.15.2 ATTRIBUTE VALUES service action**

The READ ATTRIBUTE command with ATTRIBUTE VALUES service action returns parameter data containing the attributes specified by the PARTITION NUMBER, VOLUME NUMBER, and FIRST ATTRIBUTE IDENTIFIER fields in the CDB.

The returned parameter data shall contain the requested attributes in ascending numerical order by attribute identifier value and in the format shown in table 177.

**Table 177 — READ ATTRIBUTE with ATTRIBUTE VALUES service action parameter list format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
3	AVAILABLE DATA (n-3)							(LSB)
	Attribute(s)							
4	Attribute 0 (see 7.3.1)							
	⋮							
n	Attribute x (see 7.3.1)							

The AVAILABLE DATA field shall contain the number of bytes of attribute information in the parameter list. The relationship between the AVAILABLE DATA field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The format of the attributes is described in 7.3.1.

### 6.15.3 ATTRIBUTE LIST service action

The READ ATTRIBUTE command with ATTRIBUTE LIST service action returns parameter data containing the attribute identifiers for the attributes that are not in the unsupported state and not in the nonexistent state (see 5.11) in the specified partition and volume number. The contents of FIRST ATTRIBUTE IDENTIFIER field in the CDB shall be ignored. The returned parameter data shall contain the requested attribute identifiers in ascending numerical order by attribute identifier value and in the format shown in table 178.

**Table 178 — READ ATTRIBUTE with ATTRIBUTE LIST service action parameter list format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
3	AVAILABLE DATA (n-3)							(LSB)
	Attribute identifiers							
4	ATTRIBUTE IDENTIFIER 0							
5	⋮							
n-1	ATTRIBUTE IDENTIFIER x							
n								

The AVAILABLE DATA field shall contain the number of bytes of attribute identifiers in the parameter list. The relationship between the AVAILABLE DATA field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

An ATTRIBUTE IDENTIFIER field is returned for each attribute that is not in the unsupported state and not in the non-existent state (see 5.11) in the specified partition and volume number. See 7.3.2 for a description of the attribute identifier values.

#### 6.15.4 VOLUME LIST service action

The READ ATTRIBUTE command with VOLUME LIST service action returns parameter data (see table 179) identifying the supported number of volumes. The contents of VOLUME NUMBER, PARTITION NUMBER, and FIRST ATTRIBUTE IDENTIFIER fields in the CDB shall be ignored.

**Table 179 — READ ATTRIBUTE with VOLUME LIST service action parameter list format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____							
1	AVAILABLE DATA (0002h) _____ (LSB)							
2	FIRST VOLUME NUMBER							
3	NUMBER OF VOLUMES AVAILABLE							

The AVAILABLE DATA field shall contain two. The relationship between the AVAILABLE DATA field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The FIRST VOLUME NUMBER field indicates the first volume available. Volume numbering should start at zero.

The NUMBER OF VOLUMES AVAILABLE field indicates the number of volumes available.

#### 6.15.5 PARTITION LIST service action

The READ ATTRIBUTE command with PARTITION LIST service action returns parameter data (see table 180) identifying the number of partitions supported in the specified volume number. The contents of PARTITION NUMBER and FIRST ATTRIBUTE IDENTIFIER fields in the CDB shall be ignored.

**Table 180 — READ ATTRIBUTE with PARTITION LIST service action parameter list format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____							
1	AVAILABLE DATA (0002h) _____ (LSB)							
2	FIRST PARTITION NUMBER							
3	NUMBER OF PARTITIONS AVAILABLE							

The AVAILABLE DATA field shall contain two. The relationship between the AVAILABLE DATA field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The FIRST PARTITION NUMBER field indicates the first partition available on the specified volume number. Partition numbering should start at zero.

The NUMBER OF PARTITIONS AVAILABLE field indicates the number of partitions available on the specified volume number.

## 6.16 READ BUFFER command

### 6.16.1 READ BUFFER command introduction

The READ BUFFER command (see table 181) is used in conjunction with the WRITE BUFFER command for:

- a) Testing logical unit buffer memory;
- b) Testing the integrity of the service delivery subsystem;
- c) Downloading microcode (see 5.16); and
- d) Retrieving error history.

**Table 181 — READ BUFFER command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (3Ch)							
1	Reserved			MODE				
2	BUFFER ID							
3	(MSB) <div>BUFFER OFFSET</div> (LSB)							
5								
6	(MSB) <div>ALLOCATION LENGTH</div> (LSB)							
8								
9	CONTROL							

The function of this command and the meaning of fields within the CDB depend on the contents of the MODE field. The MODE field is defined in table 182.

**Table 182 — READ BUFFER MODE field**

Code	Description	References
00h	Combined header and data <sup>a</sup>	6.16.2
01h	Vendor specific <sup>a</sup>	6.16.3
02h	Data	6.16.4
03h	Descriptor	6.16.5
04h to 09h	Reserved	
0Ah	Read data from echo buffer	6.16.6
0Bh	Echo buffer descriptor	6.16.7
0Ch to 19h	Reserved	
1Ah	Enable expander communications protocol and Echo buffer	6.16.8
1Bh	Reserved	
1Ch	Error history	5.12 and 6.16.9
1Dh to 1Fh	Reserved	
<sup>a</sup> Modes 00h and 01h are not recommended.		

If the mode is not set to one, the ALLOCATION LENGTH field is defined in 4.3.5.6.

6.16.2 Combined header and data mode (00h)

In this mode, a four-byte header followed by data bytes is returned to the application client in the Data-In Buffer. The allocation length should be set to four or greater. The BUFFER ID and the BUFFER OFFSET fields are reserved.

The four-byte READ BUFFER header (see table 183) is followed by data bytes from the buffer.

Table 183 — READ BUFFER header

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved							
1	(MSB)							
3	BUFFER CAPACITY (LSB)							
4	Data							
n								

The BUFFER CAPACITY field specifies the total number of data bytes available in the buffer. The buffer capacity is not reduced to reflect the actual number of bytes written using the WRITE BUFFER command with combined header and data mode (see 6.39.2). The relationship between the BUFFER CAPACITY field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6. Following the READ BUFFER header, the device server shall transfer data from the buffer.

NOTE 32 - The buffer is shared by all application clients and I\_T nexuses. If one application client writes the buffer, a second application client writes the buffer, and then the first application client reads the buffer, the read may or may not retrieve the data from the second application client.

6.16.3 Vendor specific mode (01h)

In this mode, the meanings of the BUFFER ID, BUFFER OFFSET, and ALLOCATION LENGTH fields are not specified by this standard.

6.16.4 Data mode (02h)

In this mode, the Data-In Buffer is filled only with logical unit buffer data. The BUFFER ID field specifies a buffer within the logical unit from which data shall be transferred. The vendor assigns buffer ID codes to buffers within the logical unit. Buffer ID zero shall be supported. If more than one buffer is supported, then additional buffer ID codes shall be assigned contiguously, beginning with one. Buffer ID code assignments for the READ BUFFER command with data mode shall be the same as for the WRITE BUFFER command with data mode (see 6.39.4). If an unsupported buffer ID code is selected, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The BUFFER OFFSET field contains the byte offset within the specified buffer from which data shall be transferred. The application client should conform to the offset boundary requirements returned in the READ BUFFER descriptor (see 6.16.5). If the device server is unable to accept the specified buffer offset, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

NOTE 33 - The buffer is shared by all application clients and I\_T nexuses. If one application client writes the buffer, a second application client writes the buffer, and then the first application client reads the buffer, the read may or may not retrieve the data from the second application client.



### 6.16.5 Descriptor mode (03h)

In this mode, a maximum of four bytes of READ BUFFER descriptor information is returned. The BUFFER OFFSET field is reserved in this mode. The allocation length should be set to four or greater. The READ BUFFER descriptor is defined as shown in table 184.

**Table 184 — READ BUFFER descriptor**

Bit Byte	7	6	5	4	3	2	1	0
0	OFFSET BOUNDARY							
1	(MSB)							
3	BUFFER CAPACITY							(LSB)

For READ BUFFER commands, the OFFSET BOUNDARY field (see table 185) applies to the following modes:

- a) Data (i.e., 02h) (see 6.16.4); and
- b) Error history (i.e., 1Ch) (see 6.16.9).

For WRITE BUFFER commands, the OFFSET BOUNDARY field (see table 185) applies to the following modes:

- a) Data (i.e., 02h) (see 6.39.4);
- b) Download microcode with offsets and activate (i.e., 06h) (see 6.39.7);
- c) Download microcode with offsets, save, and activate (i.e., 07h) (see 6.39.8); and
- d) Download microcode with offsets, save, and defer active (i.e., 0Eh) (see 6.39.10).

For data mode (i.e., 02h), the boundary alignment indicated by the OFFSET BOUNDARY field applies only to the buffer specified by the BUFFER ID field. For modes other than data to which the OFFSET BOUNDARY field applies, the boundary alignment applies regardless of the buffer specified by the BUFFER ID field.

**Table 185 — OFFSET BOUNDARY field**

Code	Description
00h to FEh	Multiples of $2^{\text{code}}$ (e.g., 00h means multiples of 1 byte or no offset restrictions, 01h means multiples of 2 bytes or even offsets, 02h means multiples of 4 bytes)
FFh	000000h is the only supported buffer offset

The BUFFER CAPACITY field indicates the maximum size in bytes of the buffer specified by the BUFFER ID field for the:

- a) READ BUFFER command with data mode (i.e., 02h); and
- b) WRITE BUFFER command with data mode (i.e., 02h).

### 6.16.6 Read data from echo buffer mode (0Ah)

In this mode the device server transfers data to the application client from the echo buffer that was written by the most recent WRITE BUFFER command with the MODE field set to write data to echo buffer (see 6.39.9) received on the same I\_T nexus. The READ BUFFER command shall return the same number of bytes of data as received in the prior WRITE BUFFER command with the mode field set to write data to echo buffer, limited by the allocation length as described in 4.3.5.6.

The BUFFER ID and BUFFER OFFSET fields are ignored in this mode.

If no WRITE BUFFER command with the mode set to write data to echo buffer received on this I\_T nexus has completed without an error, then the READ BUFFER command shall terminate with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to COMMAND SEQUENCE ERROR. If the data in the echo buffer has been overwritten by another I\_T nexus, the READ BUFFER command shall be terminated with CHECK CONDITION status, with the sense key set to ABORTED COMMAND, and the additional sense code set to ECHO BUFFER OVERWRITTEN.

After a WRITE BUFFER command with the mode set to write data to echo buffer has completed without an error, the application client may send multiple READ BUFFER commands with the mode set to read data from echo buffer in order to read the echo buffer data multiple times.

#### 6.16.7 Echo buffer descriptor mode (0Bh)

In this mode, a maximum of four bytes of READ BUFFER descriptor information is returned. The device server shall return the descriptor information for the echo buffer. If there is no echo buffer implemented, the device server shall return all zeros in the READ BUFFER descriptor. The BUFFER ID field and BUFFER OFFSET field are reserved in this mode. The allocation length should be set to four or greater. The READ BUFFER descriptor is defined as shown in table 186.

**Table 186 — Echo buffer descriptor**

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved							EBOS
1	Reserved							
2	Reserved			(MSB)				
3	BUFFER CAPACITY							(LSB)

The BUFFER CAPACITY field shall return the size of the echo buffer in bytes aligned to a four-byte boundary. The maximum echo buffer size is 4 096 bytes.

If the echo buffer is implemented, the echo buffer descriptor shall be implemented.

An echo buffer overwritten supported (EBOS) bit set to one indicates either:

- The device server returns the ECHO BUFFER OVERWRITTEN additional sense code if the data being read from the echo buffer is not the data previously written by the same I\_T nexus, or
- The device server ensures echo buffer data returned to each I\_T nexus is the same as that previously written by that I\_T nexus.

An EBOS bit set to zero specifies that the echo buffer may be overwritten by any intervening command received on any I\_T nexus.

A READ BUFFER command with the mode set to echo buffer descriptor may be used to determine the echo buffer capacity and supported features before a WRITE BUFFER command with the mode set to write data to echo buffer (see 6.39.9) is sent.

#### 6.16.8 Enable expander communications protocol and Echo buffer (1Ah)

Receipt of a READ BUFFER command with this mode (1Ah) causes a communicative expander (see SPI-5) to enter the expanded communications protocol mode. Device servers in SCSI target devices that receive a READ

BUFFER command with this mode shall process it as if it were a READ BUFFER command with mode 0Ah (see 6.16.6).

### 6.16.9 Error history mode (1Ch)

#### 6.16.9.1 Error history overview

This mode is used to manage and retrieve error history (see 5.12).

If the device server is unable to process a READ BUFFER command with the `MODE` field set to 1Ch because of a vendor specific condition, then the device server shall terminate the READ BUFFER command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to COMMAND SEQUENCE ERROR.

The BUFFER ID field (see table 187) specifies the action that the device server shall perform, and the parameter data, if any, that the device server shall return.

**Table 187 — Error history BUFFER ID field**

Code	Description	Buffer offset	Error history I_T nexus constrained	Reference
00h	Return error history directory	0000h	Yes	6.16.9.2
01h	Return error history directory and create new error history snapshot (see 3.1.49)	0000h	Yes	6.16.9.2
02h	Return error history directory and establish new error history I_T nexus (see 3.1.48)	0000h	No	6.16.9.2
03h	Return error history directory, establish new error history I_T nexus, and create new error history snapshot	0000h	No	6.16.9.2
04h to 0Fh	Reserved		Yes	
10h to EFh	Return error history	0000h to FFFFh	Yes	6.16.9.3
F0h to FDh	Reserved		Yes	
FEh	Clear error history I_T nexus	Ignored	Yes	6.16.9.4
FFh	Clear error history I_T nexus and release error history snapshot	Ignored	Yes	6.16.9.5

The command shall be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to OPERATION IN PROGRESS if the device server receives a READ BUFFER command:

- With the `MODE` field set to 1Ch;
- With the `BUFFER ID` field set to a value that table 187 shows as constrained by error history I\_T nexus;
- If an error history I\_T nexus exists and the command is received from an I\_T nexus that is different that I\_T nexus; and
- An error history snapshot exists.

The BUFFER OFFSET field specifies the byte offset from the start of the buffer specified by the `BUFFER ID` field from which the device server shall return data. The application client should conform to the offset boundary require-

ments indicated in the READ BUFFER descriptor (see 6.16.5). If the buffer offset is not one of those shown in table 187 or the device server is unable to accept the specified buffer offset, then the device server shall terminate the READ BUFFER command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

### 6.16.9.2 Error history directory

Whenever allowed by established error history I\_T nexus constraints (see 6.16.9.1), if any, all error history device server actions return an error history directory (see table 189). Some error history device server actions also discard the existing error history snapshot (see 3.1.49) and create a new error history snapshot (see table 188).

**Table 188 — Summary of error history directory device server actions**

BUFFER ID field	Establish new error history I_T nexus (see 3.1.48)	Error history snapshot (see 3.1.49)	
		Preserved (if exists)	Created
00h	No <sup>a</sup>	Yes	No <sup>b</sup>
01h	No <sup>a</sup>	No	Yes
02h	Yes	Yes	No <sup>b</sup>
03h	Yes	No	Yes
<sup>a</sup> If no error history I_T nexus is established, a new one is established.			
<sup>b</sup> If no error history snapshot exists, a new one is created.			

The error history directory is defined in table 189.

**Table 189 — Error history directory**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____							
7	T10 VENDOR IDENTIFICATION _____ (LSB)							
8	VERSION							
9	Reserved			EHS_RETRIEVED		EHS_SOURCE		CLR_SUP
10	_____							
29	Reserved _____							
30	(MSB) _____							
31	DIRECTORY LENGTH (n-31) _____ (LSB)							
	Error history directory list							
32	_____							
39	Error history directory entry [first] _____ (see table 192)							
	: _____ :							
n-7	_____							
n	Error history directory entry [last] _____ (see table 192)							

The T10 VENDOR IDENTIFICATION field contains eight bytes of left-aligned ASCII data (see 4.4.1) identifying the manufacturer of the logical unit. The T10 vendor identification shall be one assigned by INCITS. A list of assigned T10 vendor identifications is in Annex E and on the T10 web site (<http://www.t10.org>).

NOTE 34 - The T10 VENDOR IDENTIFICATION field may contain a different value than the VENDOR IDENTIFICATION field in the standard INQUIRY data (see 6.4.2) (e.g., this field may indicate a disk drive component vendor while the standard INQUIRY data indicates the original equipment manufacturer).

The VERSION field indicates the version and format of the vendor specific error history. The VERSION field is assigned by the vendor indicated in the T10 VENDOR IDENTIFICATION field.

The error history retrieved (EHS\_RETRIEVED) field (see table 190) indicates whether a clear error history device server action has been requested for the error history snapshot. EHS\_RETRIEVED field shall be set to 00b or 10b when the error history snapshot is created.

**Table 190 — EHS\_RETRIEVED field**

Code	Description
00b	No information
01b	The error history I_T nexus has requested buffer ID FEh (i.e., clear error history I_T nexus) or buffer ID FFh (i.e., clear error history I_T nexus and release snapshot) for the current error history snapshot.
10b	An error history I_T nexus has not requested buffer ID FEh (i.e., clear error history I_T nexus) or buffer ID FFh (i.e., clear error history I_T nexus and release snapshot) for the current error history snapshot.
11b	Reserved

The error history source (EHS\_SOURCE) field (see table 191) indicates the source of the error history snapshot.

**Table 191 — EHS\_SOURCE field**

Code	Description
00b	The error history snapshot was created by the device server and was not created due to processing a READ BUFFER command.
01b	Error history snapshot was created due to processing of the current READ BUFFER command
10b	Error history snapshot was created due to processing of a previous READ BUFFER command
11b	Reserved

A clear support (CLR\_SUP) bit set to one indicates that the CLR bit is supported in the WRITE BUFFER command download error history mode (see 6.39.14). A CLR\_SUP bit set to zero indicates that the CLR bit is not supported.

The DIRECTORY LENGTH field indicates the number of error history directory list bytes available to be transferred. This value shall not be altered even if the allocation length is not sufficient to transfer the entire error history directory list.

The error history directory list contains an error history directory entry (see table 192) for each supported buffer ID in the range of 00h to EFh. The first entry shall be for buffer ID 00h and the entries shall be in order of ascending buffer IDs. The supported buffer IDs are not required to be contiguous. There shall not be any entries for buffer IDs greater than or equal to F0h.

**Table 192 — Error history directory entry**

Bit Byte	7	6	5	4	3	2	1	0
0	SUPPORTED BUFFER ID							
1	Reserved							
3								
4	(MSB)	MAXIMUM AVAILABLE LENGTH						
7								(LSB)

The SUPPORTED BUFFER ID field indicates the error history buffer ID associated with this entry.

The MAXIMUM AVAILABLE LENGTH field indicates the maximum number of data bytes contained in the buffer indicated by the SUPPORTED BUFFER ID field. The actual number of bytes available for transfer may be smaller.

#### 6.16.9.3 Error history data buffer

Unless an error is encountered, the device server shall return parameter data that contains error history in a vendor specific format from the error history snapshot from the specified buffer at the specified buffer offset.

If the device server receives a READ BUFFER command with the MODE field set to 1Ch from the established error history I\_T nexus and the BUFFER ID field is set to a value that the error history directory (see 6.16.9.2) shows as not supported, then the command shall be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

If the value in the BUFFER OFFSET field is not supported, the command shall be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

The amount of error history in the specified buffer shall be less than or equal to the number of bytes indicated by the MAXIMUM AVAILABLE LENGTH field in the error history directory (see 6.16.9.2).

#### 6.16.9.4 Clear error history I\_T nexus

If the BUFFER ID field is set to FEh, the device server shall:

- a) Clear the error history I\_T nexus, if any; and
- b) Not transfer any data.

#### 6.16.9.5 Clear error history I\_T nexus and release snapshot

If the BUFFER ID field is set to FFh, the device server shall:

- a) Clear the error history I\_T nexus, if any,
- b) Release the error history snapshot, if any; and
- c) Not transfer any data.

## 6.17 READ MEDIA SERIAL NUMBER command

The READ MEDIA SERIAL NUMBER command (see table 193) reports the media serial number reported by the device and the currently mounted media.

**Table 193 — READ MEDIA SERIAL NUMBER command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (ABh)							
1	Reserved			SERVICE ACTION (01h)				
2	Reserved							
5								
6	(MSB)	ALLOCATION LENGTH						(LSB)
9								
10	Reserved							
11	CONTROL							

The ALLOCATION LENGTH field is defined in 4.3.5.6.

The READ MEDIA SERIAL NUMBER parameter data format is shown in table 194.

**Table 194 — READ MEDIA SERIAL NUMBER parameter data format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) <div>MEDIA SERIAL NUMBER LENGTH (4n-4)</div> (LSB)							
3								
4								
4n-1	MEDIA SERIAL NUMBER							

The MEDIA SERIAL NUMBER LENGTH field shall contain the number of bytes in the MEDIA SERIAL NUMBER field. The media serial number length shall be a multiple of four. The relationship between the MEDIA SERIAL NUMBER LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The MEDIA SERIAL NUMBER field shall contain the vendor specific serial number of the media currently installed. If the number of bytes in the vendor specific serial number is not a multiple of four, then up to three bytes containing zero shall be appended to the highest numbered bytes of the MEDIA SERIAL NUMBER field.

If the media serial number is not available (e.g., the currently installed media has no valid media serial number), zero shall be returned in the MEDIA SERIAL NUMBER LENGTH field.

If the media serial number is not accessible because there is no media present, the command shall be terminated with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to MEDIUM NOT PRESENT.

## 6.18 RECEIVE COPY RESULTS command

### 6.18.1 RECEIVE COPY RESULTS command introduction

The RECEIVE COPY RESULTS command (see table 195) provides a means for the application client to receive information about the copy manager or the results of a previous or current EXTENDED COPY command (see 6.3).

**Table 195 — RECEIVE COPY RESULTS command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (84h)							
1	Reserved			SERVICE ACTION				
2	LIST IDENTIFIER							
3	Reserved							
9								
10	(MSB)	ALLOCATION LENGTH						(LSB)
13								
14	Reserved							
15	CONTROL							

The service actions defined for the RECEIVE COPY RESULTS command are shown in table 196.

**Table 196 — RECEIVE COPY RESULTS service action codes**

Code	Name	Description	Returns Data While EXTENDED COPY Is In Progress	Reference
00h	COPY STATUS	Return the current copy status of the EXTENDED COPY command identified by the LIST IDENTIFIER field.	Yes	6.18.2
01h	RECEIVE DATA	Return the held data read by EXTENDED COPY command identified by the LIST IDENTIFIER field.	No	6.18.3
03h	OPERATING PARAMETERS	Return copy manager operating parameters.	Yes	6.18.4
04h	FAILED SEGMENT DETAILS	Return copy target device sense data and other information about the progress of processing a segment descriptor whose processing was not completed during processing of the EXTENDED COPY command identified by the LIST IDENTIFIER field.	No	6.18.5
05h to 1Eh	Reserved			
1Fh	Vendor Specific			



The LIST IDENTIFIER field specifies the EXTENDED COPY command (see 6.3) about which information is to be transferred. The RECEIVE COPY RESULTS command shall return information from the EXTENDED COPY command received on the same I\_T nexus with a list identifier that matches the list identifier specified in the RECEIVE COPY RESULTS CDB.

If the LIST IDENTIFIER field specifies an EXTENDED COPY command that had the NRCR bit set to one in the parameter data (see 6.3), the copy manager may respond to a RECEIVE COPY RESULTS command as if the EXTENDED COPY command had never been received.

The actual length of the RECEIVE COPY RESULTS parameter data is available in the AVAILABLE DATA parameter data field. The ALLOCATION LENGTH field is defined in 4.3.5.6. See the RECEIVE COPY RESULTS service action definitions for additional requirements.

### 6.18.2 COPY STATUS service action

In response to the COPY STATUS service action, the copy manager shall return the current status of the EXTENDED COPY command (see 6.3) specified by the LIST IDENTIFIER field in the CDB. If no EXTENDED COPY command known to the copy manager has a matching list identifier, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

Table 197 shows the format of the information returned by the copy manager in response to the COPY STATUS service action. If a device server supports the EXTENDED COPY command, it shall also support the RECEIVE COPY RESULTS command with COPY STATUS service action.

**Table 197 — Parameter data for the COPY STATUS service action**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____							
3	AVAILABLE DATA (00000008h)							(LSB) _____
4	HDD	COPY MANAGER STATUS						
5	(MSB) _____							
6	SEGMENTS PROCESSED							(LSB) _____
7	TRANSFER COUNT UNITS							
8	(MSB) _____							
11	TRANSFER COUNT							(LSB) _____

After completion of an EXTENDED COPY command, the copy manager shall preserve all data returned by a COPY STATUS service action for a vendor specific period of time. The copy manager shall discard the COPY STATUS data when:

- A RECEIVE COPY RESULTS command with COPY STATUS service action is received on the same I\_T nexus with a matching list identifier;
- When another EXTENDED COPY command is received on the same I\_T nexus and the list identifier matches the list identifier associated with the data preserved for the COPY STATUS service action;
- When the copy manager detects a logical unit reset or I\_T nexus loss; or
- When the copy manager requires the resources used to preserve the data.

The AVAILABLE DATA field shall contain the number of bytes present in the parameter data that follows. The relationship between the AVAILABLE DATA field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The held data discarded (HDD) bit indicates whether held data has been discarded. If HDD bit is set to one, held data has been discarded as described in 6.18.4. If HDD bit is set to zero, held data has not been discarded.

The COPY MANAGER STATUS field contains the current status of the EXTENDED COPY command specified by the LIST IDENTIFIER field in the CDB as defined in table 198.

**Table 198 — COPY MANAGER STATUS field**

Code	Meaning
00h	Operation in progress
01h	Operation completed without errors
02h	Operation completed with errors
03h to 7Fh	Reserved

The SEGMENTS PROCESSED field contains the number of segments the copy manager has processed for the EXTENDED COPY command specified by the LIST IDENTIFIER field in the CDB including the segment currently being processed. This field shall be zero if the copy manager has not yet begun processing segment descriptors.

The TRANSFER COUNT UNITS field specifies the units for the TRANSFER COUNT field as defined in table 199.

**Table 199 — COPY STATUS TRANSFER COUNT UNITS field**

Code	Meaning <sup>a</sup>	Multiplier to convert TRANSFER COUNT field to bytes
00h	Bytes	1
01h	Kibibytes	2 <sup>10</sup> or 1024
02h	Mebibytes	2 <sup>20</sup>
03h	Gebibytes	2 <sup>30</sup>
04h	Tebibytes	2 <sup>40</sup>
05h	Pebibytes	2 <sup>50</sup>
06h	Exbibytes	2 <sup>60</sup>
07h to FFh	Reserved	
<sup>a</sup> See 3.6.4.		

The TRANSFER COUNT field specifies the amount of data written to a destination device for the EXTENDED COPY command specified by the LIST IDENTIFIER field in the CDB prior to receiving the RECEIVE COPY RESULTS command with COPY STATUS service action.

### 6.18.3 RECEIVE DATA service action

If the copy manager supports those segment descriptors that require data to be held for transfer to the application client, then the RECEIVE DATA service action causes the copy manager to return the held data using the format shown in table 200. If a copy manager supports any of the segment descriptor type codes that require data to be held for the application client (see 6.3.5), then it shall also support the RECEIVE COPY RESULTS command with RECEIVE DATA service action.

The EXTENDED COPY command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB if any of the following conditions exist:

- a) No EXTENDED COPY command known to the copy manager has a list identifier that matches the LIST IDENTIFIER field in the CDB; or
- b) If the LIST IDENTIFIER field in the CDB identifies an EXTENDED COPY command that still is being processed by the copy manager.

**Table 200 — Parameter data for the RECEIVE DATA service action**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
3	AVAILABLE DATA (n-3)							(LSB)
4								
n	HELD DATA							

Following completion of an EXTENDED COPY command, the copy manager shall preserve all data returned by a RECEIVE DATA service action for a vendor specific period of time. The application client should issue a RECEIVE COPY RESULTS command with RECEIVE DATA service action as soon as practical following completion of the EXTENDED COPY command to insure that the data is not discarded by the copy manager. The copy manager shall discard the buffered inline data:

- a) After all data held for a specific EXTENDED COPY command has been successfully transferred to the application client;
- b) When a RECEIVE COPY RESULTS command with RECEIVE DATA service action has been received on the same I\_T nexus with a matching list identifier, with the ALLOCATION LENGTH field set to zero;
- c) When another EXTENDED COPY command is received on the same I\_T nexus and the list identifier matches the list identifier associated with the data preserved for RECEIVE DATA service action;
- d) When the copy manager detects a logical unit reset or I\_T nexus loss; or
- e) When the copy manager requires the resources used to preserve the data.

The AVAILABLE DATA field shall contain the number of bytes of held data available for delivery to the application client. The relationship between the AVAILABLE DATA field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The HELD DATA field contains the data held by the copy manager for delivery to the application client as prescribed by several segment descriptor type codes. Unless the copy manager's held data limit (see 6.18.4) is exceeded, the first byte held in response to the first segment descriptor in the EXTENDED COPY parameter list prescribing the holding of data (i.e., the oldest byte held) is returned in byte 4. The last byte held in response to the last segment descriptor in the EXTENDED COPY parameter list prescribing the holding of data (i.e., the newest byte held) is returned in byte n.

#### 6.18.4 OPERATING PARAMETERS service action

In response to the OPERATING PARAMETERS service action, the copy manager shall return its operating parameter information in the format shown in table 201. If a device server supports the EXTENDED COPY command (see 6.3), then it shall also support the RECEIVE COPY RESULTS command with OPERATING PARAMETERS service action.

**Table 201 — Parameter data for the OPERATING PARAMETERS service action**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)	AVAILABLE DATA (n-3)						(LSB)
3								
4		Reserved						
7								
8	(MSB)	MAXIMUM TARGET DESCRIPTOR COUNT						(LSB)
9								
10	(MSB)	MAXIMUM SEGMENT DESCRIPTOR COUNT						(LSB)
11								
12	(MSB)	MAXIMUM DESCRIPTOR LIST LENGTH						(LSB)
15								
16	(MSB)	MAXIMUM SEGMENT LENGTH						(LSB)
19								
20	(MSB)	MAXIMUM INLINE DATA LENGTH						(LSB)
23								
24	(MSB)	HELD DATA LIMIT						(LSB)
27								
28	(MSB)	MAXIMUM STREAM DEVICE TRANSFER SIZE						(LSB)
31								
32		Reserved						
35								
36		MAXIMUM CONCURRENT COPIES						
37		DATA SEGMENT GRANULARITY (log 2)						
38		INLINE DATA GRANULARITY (log 2)						
39		HELD DATA GRANULARITY (log 2)						
40		Reserved						
42								
43		IMPLEMENTED DESCRIPTOR LIST LENGTH (n-43)						
44		List of implemented descriptor type codes (ordered)						
n								

The AVAILABLE DATA field shall contain the number of bytes following the AVAILABLE DATA field in the parameter data (i.e., the total number of parameter data bytes minus 4). The relationship between the AVAILABLE DATA field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The MAXIMUM TARGET COUNT field contains the maximum number of target descriptors that the copy manager allows in a single EXTENDED COPY target descriptor list.

The MAXIMUM SEGMENT COUNT field contains the maximum number of segment descriptors that the copy manager allows in a single EXTENDED COPY segment descriptor list.

The MAXIMUM DESCRIPTOR LIST LENGTH field contains the maximum length, in bytes, of the target descriptor list and segment descriptor list. This length includes the embedded data but excludes inline data that follows the descriptors.

The MAXIMUM SEGMENT LENGTH field indicates the length, in bytes, of the largest amount of data that the copy manager supports writing via a single segment. Bytes introduced as a result of the PAD bit being set to one (see 6.3.7) are not counted towards this limit. A value of zero indicates that the copy manager places no limits on the amount of data written by a single segment.

The MAXIMUM INLINE DATA LENGTH field indicates the length, in bytes, of the largest amount of inline data that the copy manager supports in the EXTENDED COPY parameter list. This does not include data included as embedded data within the segment descriptors. The MAXIMUM INLINE DATA LENGTH field applies only to segment descriptors containing the 04h descriptor type code (see 6.3.7.7). The field shall be set to zero when the 04h descriptor type code is not supported by the copy manager.

The HELD DATA LIMIT field indicates the length, in bytes, of the minimum amount of data the copy manager guarantees to hold for return to the application client via the RECEIVE COPY RESULTS command with RECEIVE DATA service action (see 6.18.3). If the processing of segment descriptors requires more data to be held, the copy manager may discard some of the held data in a vendor specific manner that retains the held bytes from the most recently processed segment descriptors. The discarding of held data bytes shall not be considered an error. If held data is discarded, the HDD bit shall be set as described in 6.18.2.

The MAXIMUM CONCURRENT COPIES field contains the maximum number of EXTENDED COPY commands supported for concurrent processing by the copy manager.

The DATA SEGMENT GRANULARITY field indicates the length of the smallest data block that copy manager permits in a non-inline segment descriptor (i.e., segment descriptors with type codes other than 04h). The amount of data transferred by a single segment descriptor shall be a multiple of the granularity. The DATA SEGMENT GRANULARITY value is expressed as a power of two. Bytes introduced as a result of the PAD bit being set to one (see 6.3.7) are not counted towards the data length granularity.

The INLINE DATA GRANULARITY field indicates the length of the of the smallest block of inline data that the copy manager permits being written by a segment descriptor containing the 04h descriptor type code (see 6.3.7.7). The amount of inline data written by a single segment descriptor shall be a multiple of the granularity. The INLINE DATA GRANULARITY value is expressed as a power of two. Bytes introduced as a result of the PAD bit being set to one (see 6.3.7) are not counted towards the length granularity.

If the copy manager encounters a data or inline segment descriptor that violates either the data segment granularity or the inline data granularity, the EXTENDED COPY command shall be terminated with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to COPY SEGMENT GRANULARITY VIOLATION.

The HELD DATA GRANULARITY field indicates the length of the smallest block of held data that the copy manager shall transfer to the application client in response to a RECEIVE COPY RESULTS command with RECEIVE DATA

service action (see 6.18.3). The amount of data held by the copy manager in response to any one segment descriptor shall be a multiple of this granularity. The HELD DATA GRANULARITY value is expressed as a power of two.

The MAXIMUM STREAM DEVICE TRANSFER SIZE field indicates the maximum transfer size, in bytes, supported for stream devices.

The IMPLEMENTED DESCRIPTOR LIST LENGTH field contains the length, in bytes, of the list of implemented descriptor type codes.

The list of implemented descriptor type codes contains one byte for each segment or target DESCRIPTOR TYPE CODE value (see 6.3.5) supported by the copy manager, with a unique supported DESCRIPTOR TYPE CODE value in each byte. The DESCRIPTOR TYPE CODE values shall appear in the list in ascending numerical order.

### 6.18.5 FAILED SEGMENT DETAILS service action

In response to the FAILED SEGMENT DETAILS service action, the copy manager shall return details of the segment processing failure that caused termination of the EXTENDED COPY command (see 6.3) specified by the LIST IDENTIFIER field in the CDB. Table 202 shows the format of the information returned by the copy manager in response to a FAILED SEGMENT DETAILS service action. If a device server supports the EXTENDED COPY command (see 7.4), then it shall also support the RECEIVE COPY RESULTS command with FAILED SEGMENT DETAILS service action.

When processing of an EXTENDED COPY command is aborted and processing of a segment descriptor is incomplete, the copy manager shall preserve details about the progress in processing of that descriptor. These details enable the application client to obtain information it needs to determine the state in which copy target devices, in particular stream devices, have been left by incomplete processing.

The EXTENDED COPY command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB if any of the following conditions exist:

- a) No EXTENDED COPY command known to the copy manager has a list identifier that matches the LIST IDENTIFIER field in the CDB; or
- b) If the LIST IDENTIFIER field in the CDB identifies an EXTENDED COPY command that still is being processed by the copy manager.

**Table 202 — Parameter data for the FAILED SEGMENT DETAILS service action**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
3	AVAILABLE DATA (n-3)							(LSB)
4	Reserved							
55								
56	EXTENDED COPY COMMAND STATUS							
57	Reserved							
58	(MSB)							
59	SENSE DATA LENGTH (n-59)							(LSB)
60	SENSE DATA							
n								

The application client should issue a RECEIVE COPY RESULTS command with FAILED SEGMENT DETAILS service action immediately following failure of the EXTENDED COPY command to insure that the information is not discarded by the copy manager. The copy manager shall discard the failed segment details:

- a) After all failed segment details held for a specific EXTENDED COPY command have been successfully transferred to the application client;
- b) When a RECEIVE COPY RESULTS command with FAILED SEGMENT DETAILS service action has been received on the same I\_T nexus with a matching list identifier, with the ALLOCATION LENGTH field set to zero;
- c) When another EXTENDED COPY command is received on the same I\_T nexus using the same list identifier;
- d) When the copy manager detects a logical unit reset or I\_T nexus loss; or

- e) When the copy manager requires the resources used to preserve the data.

The AVAILABLE DATA field shall contain the number of bytes of failed segment details available for delivery to the application client. If no failed segment details data is available for the specified list identifier then the AVAILABLE DATA field shall be set to zero and no data beyond the AVAILABLE DATA field shall be returned. The relationship between the AVAILABLE DATA field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The COPY COMMAND STATUS field contains the SCSI status value that was returned for the EXTENDED COPY command identified by the LIST IDENTIFIER field in the CDB.

The SENSE DATA LENGTH field indicates how many bytes of sense data are present in the SENSE DATA field.

The SENSE DATA field contains a copy of the sense data that the copy manager prepared as part of terminating the EXTENDED COPY command identified by the list identifier with CHECK CONDITION status.

## 6.19 RECEIVE CREDENTIAL command

### 6.19.1 RECEIVE CREDENTIAL command description

#### 6.19.1.1 Overview

The RECEIVE CREDENTIAL command (see table 203) allows a secure CDB originator (see 5.14.6.2) to receive a credential from a security manager device server (e.g., a CbCS management device server (see 5.14.6.8.3)) for use in a CDB (e.g., use in the CbCS extension descriptor (see 5.14.6.8.16)).

**Table 203 — RECEIVE CREDENTIAL command**

Bit Byte	7	6	5	4	3	2	1	0						
0	OPERATION CODE (7Fh)													
1	CONTROL													
2	Reserved													
6														
7	ADDITIONAL CDB LENGTH (n-7)													
8	(MSB)	SERVICE ACTION (1800h)												
9								(LSB)						
10	(MSB)	ALLOCATION LENGTH												
11								(LSB)						
12	(MSB)	AC_SAI												
15								(LSB)						
16	ENCRYPTED REQUEST DESCRIPTOR													
n														

The ALLOCATION LENGTH field is defined in 4.3.5.6.

The AC\_SAI field contains the value of the AC\_SAI SA parameter (see 5.14.2.2) for the SA to be used to encrypt the parameter data as described in 6.19.2.1.



The ENCRYPTED REQUEST DESCRIPTOR field shall contain an ESP-SCSI CDB descriptor (see 5.14.7.4). Before encryption and after decryption, the UNENCRYPTED BYTES field (see 5.14.7.3) that are used to compute the ENCRYPTED OR AUTHENTICATED DATA field (see 5.14.7.4) contents shall have the format shown in table 204.

**Table 204 — RECEIVE CREDENTIAL command unencrypted bytes format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____							
1	CREDENTIAL REQUEST TYPE _____ (LSB)							
2	_____							
n	CREDENTIAL REQUEST DESCRIPTOR _____							

The CREDENTIAL REQUEST TYPE field (see table 205) specifies type of credential being requested and the format of the CREDENTIAL REQUEST DESCRIPTOR field.

**Table 205 — CREDENTIAL REQUEST TYPE field**

Code	Description	Reference
0001h	CbCS logical unit	6.19.1.2
0002h	CbCS logical unit and volume	6.19.1.3
all other codes	Reserved	

The CREDENTIAL REQUEST DESCRIPTOR field specifies the information needed to request the credential as described in table 205.

If return of the requested credential is not permitted, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - NO ACCESS RIGHTS.

The DS\_SAI field in the ENCRYPTED REQUEST DESCRIPTOR field contains the value of the DS\_SAI SA parameter (see 5.14.2.2) for the SA to be used to encrypt the unencrypted bytes and the parameter data as described.

If the device server is not maintaining an SA with an AC\_SAI SA parameter that matches the AC\_SAI field contents and a DS\_SAI SA parameter that matches the DS\_SAI field contents, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

If the device server is maintaining the SA specified by the AC\_SAI field and the DS\_SAI field, then the SA shall be verified for use by this RECEIVE CREDENTIAL command as follows:

- The USAGE\_TYPE SA parameter (see 5.14.2.2) shall be verified to be equal to 82h (i.e., CbCS authentication and credential encryption; and
- The USAGE\_DATA SA parameter (see 5.14.2.2) shall be verified not to contain an ALGORITHM IDENTIFIER field (see 7.6.3.6) that is set to ENCR\_NULL based on the contents the IKEv2-SCSI SAUT Cryptographic Algorithm payload (see 7.6.3.5.13) for the ENCR algorithm type (see 7.6.3.6.2) during creation of the SA (see 5.14.2.3).

If any of these SA verifications fails, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

### 6.19.1.2 CbCS logical unit credential request descriptor

If the credential request type field is set to 0001h (i.e., CbCS logical unit), then the format of the CREDENTIAL REQUEST DESCRIPTOR field is as shown in table 206.

**Table 206 — CbCS logical unit credential request descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	DESIGNATION DESCRIPTOR							
19								

The format of the DESIGNATION DESCRIPTOR field is defined in table 455 (see 7.7.3.1). The command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB if any of the fields in the DESIGNATION DESCRIPTOR field are set as follows:

- a) The DESIGNATOR TYPE field contains any value other than 3h (i.e., NAA);
- b) The ASSOCIATION field contains any value other than 00b (i.e., logical unit) or 10b (i.e., SCSI target device); or
- c) The DESIGNATOR LENGTH field is set to a value that is larger than 16.

### 6.19.1.3 CbCS logical unit and volume credential request descriptor

If the credential request type field is set to 0002h (i.e., CbCS logical unit and volume), then the format of the CREDENTIAL REQUEST DESCRIPTOR field is as shown in table 207.

**Table 207 — CbCS logical unit and volume credential request descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	DESIGNATION DESCRIPTOR							
19								
20	MAM ATTRIBUTE							
56								

The format of the DESIGNATION DESCRIPTOR field is defined in table 455 (see 7.7.3.1). The command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB if any of the fields in the DESIGNATION DESCRIPTOR field are set as follows:

- a) The DESIGNATOR TYPE field contains any value other than 3h (i.e., NAA);
- b) The ASSOCIATION field contains any value other than 00b (i.e., logical unit) or 10b (i.e., SCSI target device); or
- c) The DESIGNATOR LENGTH field is set to a value that is larger than 16.

The format of the MAM ATTRIBUTE field is defined in table 326 (see 7.3.1). If the ATTRIBUTE IDENTIFIER field in the MAM ATTRIBUTE field contains any value other than 0401h (i.e., MEDIUM SERIAL NUMBER), the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

## 6.19.2 RECEIVE CREDENTIAL parameter data

### 6.19.2.1 RECEIVE CREDENTIAL parameter data encryption

The RECEIVE CREDENTIAL parameter data shall be one of the ESP-SCSI data-in buffer descriptors shown in table 83 (see 5.14.7.5.1). The SA specified by the AC\_SAI field and the DS\_SAI field in the CDB shall be used to construct the ESP-SCSI data-in buffer descriptor as described in 5.14.7.5.

Before processing the parameter data, the application client should validate and decrypt the ESP-SCSI data-in buffer descriptor as described in 5.14.7.5. If any errors are detected by the validation and decryption processing, the parameter data should be ignored.

### 6.19.2.2 RECEIVE CREDENTIAL decrypted parameter data

Before encryption and after decryption, the UNENCRYPTED BYTES field (see 5.14.7.3) that are used to compute the ENCRYPTED OR AUTHENTICATED DATA field (see 5.14.7.5) contents shall contain a CbCS credential (see table 208).

**Table 208 — CbCS credential format**

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved				CREDENTIAL FORMAT (1h)			
1	Reserved							
2	(MSB) _____ CREDENTIAL LENGTH (n-3) _____ (LSB)							
3								
4	(MSB) _____ CAPABILITY LENGTH (k-5) _____ (LSB)							
5								
6								
k	CbCS capability descriptor (see 6.19.2.3) _____							
k+1	(MSB) _____ CAPABILITY KEY LENGTH (n-k-4)) _____ (LSB)							
k+4								
k+5								
n	CAPABILITY KEY _____							

The CREDENTIAL FORMAT field (see table 209) indicates the format of the credential.

**Table 209 — Credential format values**

Value	Description
0h	Reserved
1h	The format defined by this standard
2h to Fh	Reserved

The CREDENTIAL LENGTH field indicates the number of bytes that follow in the credential including the capability length, the CbCS capability descriptor, the capability key length, and the capability key.

The CAPABILITY LENGTH field indicates the number of bytes that follow in the capability.

The contents of the CbCS capability descriptor are defined in 6.19.2.3.

The CAPABILITY KEY LENGTH field indicates the number of bytes that follow in the capability key.

The CAPABILITY KEY field contains an integrity check value (see 3.1.72) that is computed and used as described in 5.14.6.8.12.

### 6.19.2.3 CbCS capability descriptor

#### 6.19.2.3.1 Overview

A CbCS capability descriptor (see table 210) specifies the commands that are allowed by the CbCS extension descriptor in which it appears.

**Table 210 — CbCS capability descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	DESIGNATION TYPE				KEY VERSION			
1	CBCS METHOD							
2	(MSB)	CAPABILITY EXPIRATION TIME						(LSB)
7								
8	(MSB)	INTEGRITY CHECK VALUE ALGORITHM						(LSB)
11								
12		PERMISSIONS BIT MASK						
15								
16	(MSB)	POLICY ACCESS TAG						(LSB)
19								
20		DESIGNATION DESCRIPTOR						
57								
58		DISCRIMINATOR						
71								

The DESIGNATION TYPE field (see table 211) specifies the format of the Designation descriptor.

**Table 211 — DESIGNATION TYPE field**

Code	Description	DESIGNATION DESCRIPTOR field format reference
0h	Reserved	
1h	Logical unit designation descriptor	6.19.2.3.2
2h	MAM attribute designation descriptor	6.19.2.3.3
3h to Fh	Reserved	

The KEY VERSION field specifies which working key (see 5.14.6.8.11), is being used to compute the capability key (see 5.14.6.8.12) for this CbCS capability.

The CBCS METHOD field (see table 212) specifies the CbCS method used by this CbCS capability.

**Table 212 — CBCS METHOD field**

Code	CbCS method	Reference
00h	BASIC	5.14.6.8.8.2
01h	CAPKEY	5.14.6.8.8.3
02h to EFh	Reserved	
F0h to FEh	Vendor specific	
FFh	Reserved	

The CAPABILITY EXPIRATION TIME field specifies expiration time of this CbCS capability as the number of milliseconds that have elapsed since midnight, 1 January 1970 UT. If the CAPABILITY EXPIRATION TIME field is set to zero, this CbCS capability does not have an expiration time.

If the CAPABILITY EXPIRATION TIME field is not set to zero, then:

- a) The clock maintained by the CbCS management device server (see 5.14.6.8.3) should be synchronized with the clock maintained by the enforcement manager (see 5.14.6.8.7). The method for synchronizing the clocks is outside the scope of this standard, however, the protocol should be implemented in a secure manner (e.g., it should not be possible for an adversary to set the clock in the SCSI device or in the secure CDB processor backwards to enable the reuse of expired CbCS credentials). The value in the enforcement manager's clock is available in the Current CbCS Parameters CbCS page (see 7.6.4.3.5) to assist in this synchronization;
- b) The CbCS management device server should set the CAPABILITY EXPIRATION TIME field to a value that is at least an order of magnitude larger than the allowed deviation between the clocks.

The INTEGRITY CHECK VALUE ALGORITHM field specifies the algorithm used to compute the capability key and other integrity check values for this CbCS capability. The value in the INTEGRITY CHECK VALUE ALGORITHM field is selected from the codes that the Unchangeable CbCS Parameters CbCS page (see 7.6.4.3.3) lists as supported integrity check value algorithms.

The PERMISSIONS BIT MASK field (see table 213) specifies the permissions allowed by this CbCS capability. More than one permissions bit may be set. The relationship between commands and bits in the PERMISSIONS BIT MASK field is defined in for the commands defined by this standard and in the command standard (see 3.1.27) that defines commands for a specific device type.

**Table 213 — PERMISSIONS BIT MASK field format**

Bit Byte	7	6	5	4	3	2	1	0
0	DATA READ	DATA WRITE	PARM READ	PARM WRITE	SEC MGMT	RESRV	MGMT	PHY ACC
1	Reserved							
2								
3	Restricted (see applicable command standard)							

A DATA READ bit set to zero indicates a command has no read permission for user data and protection information. A DATA READ bit set to one indicates a command has permission to read user data and protection information.

A DATA WRITE bit set to zero indicates a command has no write permission for user data and protection information. A DATA WRITE bit set to one indicates a command has permission to write user data and protection information.

A parameter data read (PARM READ) bit set to zero indicates a command has no parameter data read permission. A PARM READ bit set to one indicates a command has permission to read parameter data.

A parameter data write (PARM WRITE) bit set to zero indicates a command has no parameter data write permission. A PARM WRITE bit set to one indicates a command has permission to write parameter data.

A security management (SEC MGMT) bit set to zero indicates a command has no security management permission. A SEC MGMT bit set to one indicates a command has security management permission.

A reservation (RESRV) bit set to zero indicates a command has no persistent reservation permission. A RESRV bit set to one indicates a has permission to make or modify persistent reservations.

A management (MGMT) bit set to zero indicates a command has no storage management permission. A MGMT bit set to one indicates a command has storage management permission. Storage management is outside the scope of this standard.

A physical access (PHY ACC) bit set to zero indicates a command has no permission to affect physical access to the logical unit or volume. A PHY ACC bit set to one indicates a command has permission to affect physical access to the logical unit or volume (see SSC-3).

If the POLICY ACCESS TAG field contains a value other than zero, the policy access tag attribute of the logical unit (see 5.14.6.8.15) is compared to the POLICY ACCESS TAG field contents as part of validating the CbCS capability (see 5.14.6.8.13.2). If the POLICY ACCESS TAG field contains zero, then no comparison is made.

The DESIGNATION DESCRIPTOR field is used during the validation of the CbCS capability (see 5.14.6.8.13.2) to ensure that the command is being addressed to the correct logical unit or volume (see SSC-3). The format of the DESIGNATION DESCRIPTOR field is defined by the value in the DESIGNATION TYPE field as described in table 211.

If the CREDENTIAL REQUEST TYPE field in a RECEIVE CREDENTIAL command is set to 0001h (i.e., CbCS logical unit), then the DESIGNATION DESCRIPTOR field shall contain a logical unit designation descriptor that matches the DESIGNATION DESCRIPTOR field (see 6.19.1.2) in the CREDENTIAL REQUEST DESCRIPTOR field in the CDB. If the CREDENTIAL REQUEST TYPE field in a RECEIVE CREDENTIAL command is set to 0002h (i.e., CbCS logical unit and volume), then the DESIGNATION DESCRIPTOR field shall contain a MAM attribute designation descriptor that matches the MAM ATTRIBUTE field (see 6.19.1.3) in the CREDENTIAL REQUEST DESCRIPTOR field in the CDB.

The DISCRIMINATOR field provides uniqueness to the CbCS capability descriptor and may be used to limit the delegation or prevent leakage of the CbCS capability to other application clients. The CbCS management device server (see 5.14.6.8.3) shall not return the same CbCS capability descriptor to two secure CDB originators.

The enforcement manager (see 5.14.6.8.7) shall validate each CbCS capability descriptor it receives as described in 5.14.6.8.13.2.

### 6.19.2.3.2 Logical unit designation descriptor format

If the DESIGNATION TYPE field is set to 0001h (i.e., logical unit designation descriptor), then the format of the DESIGNATION DESCRIPTOR field is as shown in table 214.

**Table 214 — Logical unit designation descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	DESIGNATION DESCRIPTOR							
19								
20	Reserved							
37								

The format of the DESIGNATION DESCRIPTOR field is defined in table 455 (see 7.7.3.1) with the following additional requirements:

- a) The DESIGNATOR TYPE field shall contain 3h (i.e., NAA);
- b) The ASSOCIATION field shall 00b (i.e., logical unit); and
- c) The DESIGNATOR LENGTH field shall set to a value that is smaller than 17.

### 6.19.2.3.3 Volume designation descriptors

If the DESIGNATION TYPE field is set to 0002h (i.e., volume unit designation descriptor), then the format of the DESIGNATION DESCRIPTOR field is as shown in table 215.

**Table 215 — Volume designation descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	MAM ATTRIBUTE							
36								
37	Reserved							

The format of the MAM ATTRIBUTE field is defined in table 326 (see 7.3.1) with the following additional requirements:

- a) The MAM ATTRIBUTE field shall contain 0401h (i.e., MEDIUM SERIAL NUMBER); and
- b) The attribute length field shall contain 0020h.

## 6.20 RECEIVE DIAGNOSTIC RESULTS command

The RECEIVE DIAGNOSTIC RESULTS command (see table 216) requests that data be sent to the application client Data-In Buffer. The data is either data based on the most recent SEND DIAGNOSTIC command (see 6.32) or is a diagnostic page specified by the PAGE CODE field.

A page code valid (PCV) bit set to zero specifies that the device server return parameter data based on the most recent SEND DIAGNOSTIC command (e.g., the diagnostic page with the same page code as that specified in the

**Table 216 — RECEIVE DIAGNOSTIC RESULTS command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (1Ch)							
1	Reserved							PCV
2	PAGE CODE							
3	(MSB)	ALLOCATION LENGTH						(LSB)
4								
5	CONTROL							

most recent SEND DIAGNOSTIC command). The response to a RECEIVE DIAGNOSTIC RESULTS command with the PCV bit set to zero is vendor specific if:

- The most recent SEND DIAGNOSTIC command was not a SEND DIAGNOSTIC command defining parameter data to return;
- A RECEIVE DIAGNOSTIC RESULTS command with a PCV bit set to one has been processed since the last SEND DIAGNOSTIC command was processed; or
- No SEND DIAGNOSTIC command defining parameter data to return has been processed since power on, hard reset, or logical unit reset.

A page code valid (PCV) bit set to one specifies that the device server return the diagnostic page specified in the PAGE CODE field. Page code values are defined in 7.1 or in another command standard (see 3.1.27).

#### NOTES

- Logical units compliant with previous versions of this standard (e.g., SPC-2) may transfer more than one diagnostic page in the parameter data if the PCV bit is set to zero and the previous SEND DIAGNOSTIC command sent more than one diagnostic page in the parameter list.
- To ensure that the diagnostic command information is not destroyed by a command sent from another I\_T nexus, the logical unit should be reserved.
- Although diagnostic software is generally device-specific, this command and the SEND DIAGNOSTIC command provide a means to isolate the operating system software from the device-specific diagnostic software. The operating system may remain device-independent.

The ALLOCATION LENGTH field is defined in 4.3.5.6.

See 7.1 for RECEIVE DIAGNOSTIC RESULTS diagnostic page format definitions.

## 6.21 REPORT ALIASES command

The REPORT ALIASES command (see table 217) requests that the device server return the alias list. The alias list is managed using the CHANGE ALIASES command (see 6.2). If the CHANGE ALIASES command is supported, the REPORT ALIASES command shall also be supported.

The REPORT ALIASES command is a service action of the MAINTENANCE IN command. Additional MAINTENANCE IN service actions are defined in SCC-2 and in this standard. The MAINTENANCE IN service actions defined in SCC-2 apply only to logical units that return a device type of 0Ch or the SCCS bit set to one in their standard INQUIRY data (see 6.4.2).

The ALLOCATION LENGTH field is defined in 4.3.5.6.



**Table 217 — REPORT ALIASES command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A3h)							
1	Reserved			SERVICE ACTION (0Bh)				
2	Reserved							
5								
6	(MSB)							
9	ALLOCATION LENGTH							
	(LSB)							
10	Reserved							
11	CONTROL							

The parameter data returned by a REPORT ALIASES command (see table 218) contains zero or more alias entries.

**Table 218 — REPORT ALIASES parameter data**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
3	ADDITIONAL LENGTH (n-3)							
4	(LSB)							
4	Reserved							
5	Reserved							
6	NUMBER OF ALIASES (X)							
7								
	Alias entry (or entries)							
8	Alias entry 0 (see 6.2.2)							
	:							
	:							
	:							
n	Alias entry x (see 6.2.2)							

The ADDITIONAL LENGTH field indicates the number of bytes in the remaining parameter data. The relationship between the ADDITIONAL LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The NUMBER OF ALIASES field indicates the number of alias entries in the alias list and shall not be changed if the CDB contains an insufficient allocation length.

The parameter data shall include one alias entry for each alias in the alias list. The format of an alias entry is described in 6.2.2.

## 6.22 REPORT IDENTIFYING INFORMATION command

### 6.22.1 REPORT IDENTIFYING INFORMATION command overview

The REPORT IDENTIFYING INFORMATION command (see table 219) requests that the device server send identifying information (see 5.15) to the application client. The REPORT IDENTIFYING INFORMATION command is an extension to the REPORT PERIPHERAL DEVICE/COMPONENT DEVICE IDENTIFIER service action of the MAINTENANCE IN command defined in SCC-2. Additional MAINTENANCE IN and MAINTENANCE OUT service actions are defined in SCC-2 and in this standard.

The MAINTENANCE IN service actions defined only in SCC-2 shall apply only to logical units that return a device type of 0Ch (i.e., storage array controller device) or the SCCS bit set to one in their standard INQUIRY data (see 6.4.2). When a logical unit returns a device type of 0Ch or the SCCS bit set to one in its standard INQUIRY data, the implementation requirements for the SCC-2 MAINTENANCE IN service actions shall be as specified in SCC-2. Otherwise the MAINTENANCE IN service action definitions and implementation requirements stated in this standard shall apply.

The device server shall return the same identifying information regardless of the I\_T nexus being used to retrieve the identifying information.

Processing a REPORT DEVICE IDENTIFYING INFORMATION command may require the enabling of a nonvolatile memory within the logical unit. If the nonvolatile memory is not ready, the command shall be terminated with CHECK CONDITION status, and not wait for the nonvolatile memory to become ready. The sense key shall be set to NOT READY and the additional sense code shall be set as described in table 270 (see 6.37). This information should allow the application client to determine the action required to cause the device server to become ready.

**Table 219 — REPORT IDENTIFYING INFORMATION command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A3h)							
1	Reserved			SERVICE ACTION (05h)				
2	Reserved							
3								
4	Restricted (see SCC-2)							
5								
6	(MSB) ALLOCATION LENGTH (LSB)							
9								
10	INFORMATION TYPE							Reserved
11	CONTROL							

The ALLOCATION LENGTH field is defined in 4.3.5.6.

The INFORMATION TYPE field (see table 220) specifies the type of information to be reported.

**Table 220 — INFORMATION TYPE field**

Code	Description	Reference
0000000b	Peripheral device identifying information (see 5.15).	6.22.2
0000010b	Peripheral device text identifying information (see 5.15).	6.22.2
1111111b	Identifying information supported – The parameter data contains a list of supported identifying information types and the maximum length of each.	6.22.3
xxxxxx1b	Restricted	SCC-2
All other	Reserved	

### 6.22.2 IDENTIFYING INFORMATION parameter data

The REPORT IDENTIFYING INFORMATION parameter data format used when the INFORMATION TYPE field is set to 0000000b or 0000010b is shown in table 221.

**Table 221 — REPORT IDENTIFYING INFORMATION parameter data**

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved							
1								
2	(MSB)	INFORMATION LENGTH (n-3)						
3								(LSB)
4	INFORMATION							
n								

The INFORMATION LENGTH field indicates the length in bytes of the INFORMATION field. The relationship between the INFORMATION LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The INFORMATION field contains the identifying information that has the specified information type (see 6.22.1).

### 6.22.3 IDENTIFYING INFORMATION SUPPORTED parameter data

The REPORT IDENTIFYING INFORMATION parameter data format used when the INFORMATION TYPE field is set to 1111111b is shown in table 222.

**Table 222 — REPORT IDENTIFYING INFORMATION SUPPORTED parameter data**

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved							
1								
2	(MSB)	IDENTIFYING INFORMATION LENGTH (n-3)						
3								(LSB)
	Identifying information descriptor list							
4	Identifying information descriptor [first] (see table 223)							
7								
	⋮							
n-3	Identifying information descriptor [last] (see table 223)							
n								

The IDENTIFYING INFORMATION LENGTH field indicates the length in bytes of the identifying information descriptor list. The relationship between the IDENTIFYING INFORMATION LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The identifying information descriptor list contains an identifying information descriptor (see table 223) for each identifying information type supported by the device server. The identifying information descriptors shall be sorted in increasing order by information type.

**Table 223 — Identifying information descriptor**

Bit Byte	7	6	5	4	3	2	1	0
0	INFORMATION TYPE							Reserved
1	Reserved							
2	(MSB)	MAXIMUM INFORMATION LENGTH						
3								(LSB)

The INFORMATION TYPE field indicates the information type (see 5.15).

The MAXIMUM INFORMATION LENGTH field indicates the maximum number of bytes supported for identifying information that has the indicated information type (see 5.15).

## 6.23 REPORT LUNS command

The REPORT LUNS command (see table 224) requests that the peripheral device logical unit inventory accessible to the I\_T nexus be sent to the application client. The logical unit inventory is a list that shall include the logical unit numbers of all logical units having a PERIPHERAL QUALIFIER value of 000b (see 6.4.2). Logical unit numbers for logical units with PERIPHERAL QUALIFIER values other than 000b and 011b may be included in the logical unit inventory. Logical unit numbers for logical units with a PERIPHERAL QUALIFIER value of 011b shall not be included in the logical unit inventory.

**Table 224 — REPORT LUNS command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A0h)							
1	Reserved							
2	SELECT REPORT							
3	Reserved							
5								
6	(MSB)	ALLOCATION LENGTH						
9								(LSB)
10	Reserved							
11	CONTROL							

The SELECT REPORT field (see table 225) specifies the types of logical unit addresses that shall be reported.

**Table 225 — SELECT REPORT field**

Code	Description
00h	The list shall contain the logical units accessible to the I_T nexus with the following addressing methods (see SAM-4): a) Logical unit addressing method, b) Peripheral device addressing method; and c) Flat space addressing method. If there are no logical units, the LUN LIST LENGTH field shall be zero.
01h	The list shall contain only well known logical units, if any. If there are no well known logical units, the LUN LIST LENGTH field shall be zero.
02h	The list shall contain all logical units accessible to the I_T nexus.
03h to FFh	Reserved

The ALLOCATION LENGTH field is defined in 4.3.5.6. The allocation length should be at least 16.

NOTE 38 - Device servers compliant with SPC return CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB when the allocation length is less than 16 bytes.

The REPORT LUNS command shall return CHECK CONDITION status only when the device server is unable to return the requested report of the logical unit inventory.

If a REPORT LUNS command is received from an I\_T nexus with a pending unit attention condition (i.e., before the device server reports CHECK CONDITION status), the device server shall perform the REPORT LUNS command (see SAM-4).

The REPORT LUNS parameter data should be returned even though the device server is not ready for other commands. The report of the logical unit inventory should be available without incurring any media access delays. If the device server is not ready with the logical unit inventory or if the inventory list is null for the requesting I\_T nexus and the SELECT REPORT field set to 02h, then the device server shall provide a default logical unit inventory that contains at least LUN 0 or the REPORT LUNS well known logical unit (see 8.2). A non-empty peripheral device logical unit inventory that does not contain either LUN 0 or the REPORT LUNS well known logical unit is valid.

If a REPORT LUNS command is received for a logical unit that the SCSI target device does not support and the device server is not capable of returning the logical unit inventory, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to LOGICAL UNIT NOT SUPPORTED.

The device server shall report those devices in the logical unit inventory using the format shown in table 226.

Table 226 — REPORT LUNS parameter data format

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
3	LUN LIST LENGTH (n-7)							(LSB)
4	Reserved							
7								
	LUN list							
8	LUN [first]							
15								
	⋮							
n-7	LUN [last]							
n								

The LUN LIST LENGTH field shall contain the length in bytes of the LUN list that is available to be transferred. The LUN list length is the number of logical unit numbers in the logical unit inventory multiplied by eight. The relationship between the LUN LIST LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

## 6.24 REPORT PRIORITY command

The REPORT PRIORITY command (see table 227) requests the priority that has been assigned to one or more I\_T nexuses associated with the logical unit (i.e., I\_T\_L nexuses).

The REPORT PRIORITY command is a service action of the MAINTENANCE IN command. Additional MAINTENANCE IN service actions are defined in SCC-2 and in this standard. The MAINTENANCE IN service actions defined in SCC-2 apply only to logical units that return a device type of 0Ch or the sccs bit set to one in their standard INQUIRY data (see 6.4.2).

**Table 227 — REPORT PRIORITY command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A3h)							
1	Reserved			SERVICE ACTION (0Eh)				
2	PRIORITY REPORTED		Reserved					
3	Reserved							
5								
6	(MSB)							
9	ALLOCATION LENGTH							
	(LSB)							
10	Reserved							
11	CONTROL							

The PRIORITY REPORTED field (see table 228) specifies the information to be returned in the parameter data.

**Table 228 — PRIORITY REPORTED field**

Code	Description
00b	Only the priority for the I_T nexus on which the command was received shall be reported in the REPORT PRIORITY parameter data.
01b	The priority for each I_T nexus that is not set to the initial command priority shall be reported in the REPORT PRIORITY parameter data.
10b to 11b	Reserved

The ALLOCATION LENGTH field is defined in 4.3.5.6. The allocation length should be at least four.

The format of the parameter data returned by the REPORT PRIORITY command is shown in table 229.

**Table 229 — REPORT PRIORITY parameter data format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
3	PRIORITY PARAMETER DATA LENGTH (n-3)							(LSB)
	Priority descriptors							
4	Priority descriptor [first] (see table 230)							
	⋮							
n	Priority descriptor [last] (see table 230)							

The PRIORITY PARAMETER DATA LENGTH field indicates the number of bytes of parameter data that follow.

Each priority descriptor (see table 230) contains priority information for a single I\_T\_L nexus.

**Table 230 — Priority descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved				CURRENT PRIORITY			
1	Reserved							
2	(MSB)							
3	RELATIVE TARGET PORT IDENTIFIER							(LSB)
4	Reserved							
5	Reserved							
6	(MSB)							
7	ADDITIONAL DESCRIPTOR LENGTH (n-7)							(LSB)
8	TRANSPORTID							
n								

The CURRENT PRIORITY field contains the priority assigned to the I\_T\_L nexus represented by this descriptor. If the PRIORITY REPORTED field in this command is set to 00b and the priority for the I\_T\_L nexus associated with this command is set to the initial command priority, then the CURRENT PRIORITY field shall be set to zero. The priority assigned to an I\_T\_L nexus may be used as a command priority for tasks received via that I\_T\_L nexus (see SAM-4).

The RELATIVE TARGET PORT IDENTIFIER field contains the relative port identifier (see 3.1.120) of the target port that is part of the I\_T\_L nexus to which the current priority applies.

The ADDITIONAL DESCRIPTOR LENGTH field indicates the number of bytes that follow in the descriptor (i.e., the size of the TransportID).



The TRANSPORTID field contains a TransportID (see 7.5.4) identifying the initiator port that is part of the I\_T\_L nexus to which the current priority applies.

## 6.25 REPORT SUPPORTED OPERATION CODES command

### 6.25.1 REPORT SUPPORTED OPERATION CODES command introduction

The REPORT SUPPORTED OPERATION CODES command (see table 231) requests information on commands the addressed logical unit supports. An application client may request a list of all operation codes and service actions supported by the logical unit or the command support data for a specific command.

The REPORT SUPPORTED OPERATION CODES command is a service action of the MAINTENANCE IN command. Additional MAINTENANCE IN service actions are defined in SCC-2 and in this standard. The MAINTENANCE IN service actions defined in SCC-2 apply only to logical units that return a device type of 0Ch or the sccs bit set to one in their standard INQUIRY data (see 6.4.2).

**Table 231 — REPORT SUPPORTED OPERATION CODES command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A3h)							
1	Reserved			SERVICE ACTION (0Ch)				
2	RCTD	Reserved			REPORTING OPTIONS			
3	REQUESTED OPERATION CODE							
4	(MSB)	REQUESTED SERVICE ACTION						(LSB)
5								
6	(MSB)	ALLOCATION LENGTH						(LSB)
9								
10	Reserved							
11	CONTROL							

A return command timeouts descriptor (RCTD) bit set to one specifies that the command timeouts descriptor (see 6.25.4) shall be included in each command descriptor (see 6.25.2) that is returned or in the one\_command parameter data (see 6.25.3) that is returned. A RCTD bit set to zero specifies that the command timeouts descriptor shall not be included in any parameter data returned.

The REPORTING OPTIONS field (see table 232) specifies the information to be returned in the parameter data.

**Table 232 — REPORT SUPPORTED OPERATION CODES REPORTING OPTIONS field**

Code	Description	Parameter Data Reference
000b	A list of all operation codes and service actions supported by the logical unit shall be returned in the all_commands parameter data format. The REQUESTED OPERATION CODE CDB field and REQUESTED SERVICE ACTION CDB field shall be ignored.	6.25.2
001b	The command support data for the operation code specified in the REQUESTED OPERATION CODE field shall be returned in the one_command parameter data format. The REQUESTED SERVICE ACTION CDB field shall be ignored. If the REQUESTED OPERATION CODE field specifies an operation code that has service actions, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.	6.25.3
010b	The command support data for the operation code and service action specified in the REQUESTED OPERATION CODE CDB field and REQUESTED SERVICE ACTION CDB field shall be returned in the one_command parameter data format. If the REQUESTED OPERATION CODE CDB field specifies an operation code that does not have service actions, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.	6.25.3
011b to 111b	Reserved	

The REQUESTED OPERATION CODE field specifies the operation code of the command to be returned in the one\_command parameter data format (see 6.25.3).

The REQUESTED SERVICE ACTION field specifies the service action of the command to be returned in the one\_command parameter data format.

The ALLOCATION LENGTH field is defined in 4.3.5.6.

### 6.25.2 All\_commands parameter data format

The REPORT SUPPORTED OPERATION CODES all\_commands parameter data format (see table 233) begins with a four-byte header that contains the length in bytes of the parameter data followed by a list of supported commands. Each command descriptor contains information about a single supported command CDB (i.e., one operation code and service action combination, or one non-service-action operation code). The list of command descriptors shall contain all commands supported by the logical unit.

**Table 233 — All\_commands parameter data**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
3	COMMAND DATA LENGTH (n-3)							(LSB)
	Command descriptors							
4	Command descriptor 0 (see table 234)							
	⋮							
n	Command descriptor x (see table 234)							

The COMMAND DATA LENGTH field indicates the length in bytes of the command descriptor list.

Each command descriptor (see table 234) contains information about a single supported command CDB.

**Table 234 — Command descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	Reserved							
2	(MSB)							
3	SERVICE ACTION							(LSB)
4	Reserved							
5	Reserved						CTDP	SERVACTV
6	(MSB)							
7	CDB LENGTH							(LSB)
8	Command timeouts descriptor, if any (see 6.25.4)							
19								

The OPERATION CODE field contains the operation code of a command supported by the logical unit.

The SERVICE ACTION field contains a supported service action of the supported operation code indicated by the OPERATION CODE field. If the operation code indicated in the OPERATION CODE field does not have a service actions, the SERVICE ACTION field shall be set to 00h.

A command timeouts descriptor present (CTDP) bit set to one indicates that the command timeouts descriptor (see 6.25.4) is included in this command descriptor. A CTDP bit set to zero indicates that the command timeouts descriptor is not included in this command descriptor.

A service action valid (SERVACTV) bit set to zero indicates the operation code indicated by the OPERATION CODE field does not have service actions and the SERVICE ACTION field contents are reserved. A SERVACTV bit set to one indicates the operation code indicated by the OPERATION CODE field has service actions and the contents of the SERVICE ACTION field are valid.

The CDB LENGTH field contains the length of the command CDB in bytes for the operation code indicated in the OPERATION CODE field, and if the SERVACTV bit is set to the service action indicated by the SERVICE ACTION field.

If the RCTD bit is set to one in the REPORT SUPPORTED OPERATION CODES CDB (see 6.25.1), the command timeouts descriptor (see table 237 in 6.25.4) shall be included. If the RCTD bit is set to zero, the command timeouts descriptor shall not be included.

6.25.3 One\_command parameter data format

The REPORT SUPPORTED OPERATION CODES one\_command parameter data format (see table 235) contains information about the CDB and a usage map for bits in the CDB for the command specified by the REPORTING OPTIONS, REQUESTED OPERATION CODE, and REQUESTED SERVICE ACTION fields in the REPORT SUPPORTED OPERATION CODES CDB.

Table 235 — One\_command parameter data

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved							
1	CTDP	Reserved				SUPPORT		
2	(MSB)							
3	CDB SIZE (n-3)							(LSB)
4								
n	CDB USAGE DATA							
n+1								
n+12	Command timeouts descriptor, if any (see 6.25.4)							

A command timeouts descriptor present (CTDP) bit set to one indicates that the command timeouts descriptor (see 6.25.4) is included in the parameter data. A CTDP bit set to zero indicates that the command timeouts descriptor is not included in the parameter data.

The SUPPORT field is defined in table 236.

**Table 236 — SUPPORT values**

Support	Description
000b	Data about the requested SCSI command is not currently available. All data after byte 1 is not valid. A subsequent request for command support data may be successful.
001b	The device server does not support the requested command. All data after byte 1 is undefined.
010b	Reserved
011b	The device server supports the requested command in conformance with a SCSI standard. The parameter data format conforms to the definition in table 235.
100b	Reserved
101b	The device server supports the requested command in a vendor specific manner. The parameter data format conforms to the definition in table 235.
110b to 111b	Reserved

The CDB SIZE field contains the size of the CDB USAGE DATA field in the parameter data, and the number of bytes in the CDB for command being queried (i.e., the command specified by the REPORTING OPTIONS, REQUESTED OPERATION CODE, and REQUESTED SERVICE ACTION fields in the REPORT SUPPORTED OPERATION CODES CDB).

The CDB USAGE DATA field contains information about the CDB for the command being queried. The first byte of the CDB USAGE DATA field shall contain the operation code for the command being queried. If the command being queried contains a service action, then that service action code shall be placed in the CDB USAGE DATA field in the same location as the SERVICE ACTION field of the command CDB. All other bytes of the CDB USAGE DATA field shall contain a usage map for bits in the CDB for the command being queried.

The bits in the usage map shall have a one-for-one correspondence to the CDB for the command being queried. If the device server evaluates a bit in the CDB for the command being queried, the usage map shall contain a one in the corresponding bit position. If any bit representing part of a field is returned as one, all bits for the field shall be returned as one. If the device server ignores or treats as reserved a bit in the CDB for the command being queried, the usage map shall contain a zero in the corresponding bit position. The usage map bits for a given CDB field all shall have the same value.

For example, the CDB usage bit map for the REPORT SUPPORTED OPERATION CODES command is: A3h, 0Ch, 87h, FFh, FFh, FFh, FFh, FFh, FFh, 00h, 07h. This example assumes that the logical unit only supports the low-order three bits of the CDB CONTROL byte. The first byte contains the operation code, and the second byte contains three reserved bits and the service action. The remaining bytes contain the usage map.

If the RCTD bit is set to one in the REPORT SUPPORTED OPERATION CODES CDB (see 6.25.1), the command timeouts descriptor (see table 237 in 6.25.4) shall be included. If the RCTD bit is set to zero, the command timeouts descriptor shall not be included.

## 6.25.4 Command timeouts descriptor

### 6.25.4.1 Overview

The command timeouts descriptor (see table 237) returns timeout information for commands supported by the logical unit based on the time from the start of processing for the command to its reported completion.

Values returned in the command timeouts descriptor do not include times that are outside the control of the device server (e.g., prior commands with the IMMED bit set to one in the CDB, concurrent commands from the same or different I\_T nexuses, manual unloads, power-on self tests, prior aborted commands, commands that force cache synchronization, delays in the service delivery subsystem).

For commands that cause a change in power condition (see 5.10), values returned in the command timeouts descriptor do not include the power condition transition time (e.g., the time to spinup rotating media).

Values returned in the command timeouts descriptor should not be used to compare products.

**Table 237 — Command timeouts descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____							
1	DESCRIPTOR LENGTH (0Ah) _____ (LSB)							
2	Reserved							
3	COMMAND SPECIFIC							
4	(MSB) _____							
7	NOMINAL COMMAND PROCESSING TIMEOUT _____ (LSB)							
8	(MSB) _____							
11	RECOMMENDED COMMAND TIMEOUT _____ (LSB)							

The DESCRIPTOR LENGTH field indicates the number of bytes that follow in the command timeouts descriptor.

The COMMAND SPECIFIC field contains timeout information (see table 238) that is specific to one or more commands. If no command specific timeout information is defined by this or the applicable command standard (see 3.1.27) the COMMAND SPECIFIC field is reserved.

**Table 238 — Command timeouts descriptor COMMAND SPECIFIC field usage in this standard**

Command	Reference
WRITE BUFFER	6.25.4.2

A non-zero value in the NOMINAL COMMAND PROCESSING TIMEOUT field indicates the minimum amount of time in seconds the application client should wait prior to querying for the progress of the command identified by the parameter data that contains this command timeouts descriptor. A value of zero in the NOMINAL COMMAND PROCESSING TIMEOUT field indicates that no timeout is indicated.

NOTE 39 - The value contained in the NOMINAL COMMAND PROCESSING TIMEOUT field may include time required for typical device error recovery procedures expected to occur on a regular basis.

A non-zero value in the RECOMMENDED COMMAND TIMEOUT field specifies the recommended time in seconds the application client should wait prior to timing out the command identified by the parameter data that contains this command timeouts descriptor. A value of zero in the RECOMMENDED COMMAND TIMEOUT field indicates that no time is indicated.

The device server should set the recommended command timeout to a value greater than or equal to the nominal command processing timeout.

#### 6.25.4.2 WRITE BUFFER command timeouts descriptor COMMAND SPECIFIC field usage

For the WRITE BUFFER command (see 6.39), the COMMAND SPECIFIC field usage is reserved for all modes except the following:

- a) Download microcode mode (04h);
- b) Download microcode and save mode (05h);
- c) Download microcode with offsets mode (06h);
- d) Download microcode with offsets and save mode (07h);
- e) Download microcode with offsets and defer activation mode (0Eh) only if the microcode is activated by an event other than an activate deferred microcode mode; and
- f) Activate deferred microcode mode (0Fh).

If the command timeouts descriptor describes one of the WRITE BUFFER modes listed in this subclause, then the COMMAND SPECIFIC field indicates the maximum time, in one second increments, that access to the SCSI device is limited or not possible through any SCSI ports associated with a logical unit that processes a WRITE BUFFER command that specifies one of the named modes. A value of zero in the COMMAND SPECIFIC field indicates that the no maximum time is indicated.

### 6.26 REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS command

The REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS command (see table 239) requests information on task management functions (see SAM-4) the addressed logical unit supports.

The REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS command is a service action of the MAINTENANCE IN command. Additional MAINTENANCE IN service actions are defined in SCC-2 and in this standard. The MAINTENANCE IN service actions defined in SCC-2 apply only to logical units that return a device type of 0Ch or the SCCS bit set to one in their standard INQUIRY data (see 6.4.2).

**Table 239 — REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A3h)							
1	Reserved			SERVICE ACTION (0Dh)				
2	Reserved							
5								
6	(MSB)							
9	ALLOCATION LENGTH (4h or larger)							(LSB)
10	Reserved							
11	CONTROL							

The ALLOCATION LENGTH field specifies the number of bytes that have been allocated for the returned parameter data. The allocation length should be at least four. If the allocation length is less than four, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The format of the parameter data returned by the REPORT TASK MANAGEMENT FUNCTIONS command is shown in table 240.

**Table 240 — REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS parameter data**

Bit Byte	7	6	5	4	3	2	1	0
0	ATS	ATSS	CACAS	CTSS	LURS	QTS	TRS	WAKES
1	Reserved					QAES	QTSS	ITNRS
2	Reserved							
3								

An ABORT TASK supported (ATS) bit set to one indicates the ABORT TASK task management function (see SAM-4) is supported by the logical unit. An ATS bit set to zero indicates the ABORT TASK task management function is not supported.

An ABORT TASK SET supported (ATSS) bit set to one indicates the ABORT TASK SET task management function (see SAM-4) is supported by the logical unit. An ATSS bit set to zero indicates the ABORT TASK SET task management function is not supported.

A CLEAR ACA supported (CACAS) bit set to one indicates the CLEAR ACA task management function (see SAM-4) is supported by the logical unit. An CACAS bit set to zero indicates the CLEAR ACA task management function is not supported.

A CLEAR TASK SET supported (CTSS) bit set to one indicates the CLEAR TASK SET task management function (see SAM-4) is supported by the logical unit. An CTSS bit set to zero indicates the CLEAR TASK SET task management function is not supported.

A LOGICAL UNIT RESET supported (LURS) bit set to one indicates the LOGICAL UNIT RESET task management function (see SAM-4) is supported by the logical unit. An LURS bit set to zero indicates the LOGICAL UNIT RESET task management function is not supported.

A QUERY TASK supported (QTS) bit set to one indicates the QUERY TASK task management function (see SAM-4) is supported by the logical unit. An QTS bit set to zero indicates the QUERY TASK task management function is not supported.

A TARGET RESET supported (TRS) bit set to one indicates the TARGET RESET task management function (see SAM-2) is supported by the logical unit. An TRS bit set to zero indicates the TARGET RESET task management function is not supported.

A WAKEUP supported (WAKES) bit set to one indicates the WAKEUP task management function (see SAM-2) is supported by the logical unit. An WAKES bit set to zero indicates the WAKEUP task management function is not supported.

A QUERY ASYNCHRONOUS EVENT supported (QAES) bit set to one indicates the QUERY ASYNCHRONOUS EVENT task management function (see SAM-4) is supported by the logical unit. A QAES bit set to zero indicates the QUERY ASYNCHRONOUS EVENT task management function is not supported.

A QUERY TASK SET supported (QTSS) bit set to one indicates the QUERY TASK SET task management function (see SAM-4) is supported by the logical unit. A QTSS bit set to zero indicates the QUERY TASK SET task management function is not supported.



An I\_T NEXUS RESET supported (ITNRS) bit set to one indicates the I\_T NEXUS RESET task management function (see SAM-4) is supported by the logical unit. An ITNRS bit set to zero indicates the I\_T NEXUS RESET task management function is not supported.

## 6.27 REPORT TARGET PORT GROUPS command

The REPORT TARGET PORT GROUPS command (see table 241) requests that the device server send target port group information to the application client. This command shall be supported by logical units that report in the standard INQUIRY data (see 6.4.2) that they support asymmetric logical unit access (i.e., return a non-zero value in the TPGS field).

The REPORT TARGET PORT GROUPS command is a service action of the MAINTENANCE IN command. Additional MAINTENANCE IN service actions are defined in SCC-2 and in this standard. The MAINTENANCE IN service actions defined only in SCC-2 apply only to logical units that return a device type of 0Ch or the sccs bit set to one in their standard INQUIRY data.

**Table 241 — REPORT TARGET PORT GROUPS command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A3h)							
1	Reserved			SERVICE ACTION (0Ah)				
2	Reserved							
5								
6	(MSB)							
9	ALLOCATION LENGTH							
10	(LSB)							
11	Reserved							
	CONTROL							

The ALLOCATION LENGTH field is defined in 4.3.5.6.

Returning REPORT TARGET PORT GROUPS parameter data may require the enabling of a nonvolatile memory. If the nonvolatile memory is not ready, the command shall be terminated with CHECK CONDITION status, rather than wait for the nonvolatile memory to become ready. The sense key shall be set to NOT READY and the additional sense code shall be set as described in table 270 (see 6.37).

The format for the parameter data returned by the REPORT TARGET PORT GROUPS command is shown in table 242.

**Table 242 — REPORT TARGET PORT GROUPS parameter data format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
3	RETURN DATA LENGTH (n-3) (LSB)							
	Target port group descriptor(s)							
4	Target port group descriptor [first] (see table 243)							
	⋮							
n	Port group descriptor [last] (see table 243)							

The RETURN DATA LENGTH field indicates the length in bytes of the list of target port groups. The relationship between the RETURN DATA LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

There shall be one target port group descriptor (see table 243) for each target port group.

**Table 243 — Target port group descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	PREF	Reserved			ASYMMETRIC ACCESS STATE			
1	T_SUP	O_SUP	Reserved		U_SUP	S_SUP	AN_SUP	AO_SUP
2	(MSB)							
3	TARGET PORT GROUP (LSB)							
4	Reserved							
5	STATUS CODE							
6	Vendor specific							
7	TARGET PORT COUNT							
	Target port descriptor(s)							
8	Target port descriptor [first] (see table 246)							
11	⋮							
n-3	Port group descriptor [last] (see table 246)							
n								

A preferred target port (PREF) bit set to one indicates that the primary target port group is a preferred primary target port group for accessing the addressed logical unit (see 5.9.2.6). A PREF bit set to zero indicates the primary target port group is not a preferred primary target port group.

The ASYMMETRIC ACCESS STATE field (see table 244) contains the target port group's current target port asymmetric access state.

**Table 244 — ASYMMETRIC ACCESS STATE field**

Code	State (see 5.9.2.4)	Type (see 5.9.2.1)
0h	Active/optimized	Primary
1h	Active/non-optimized	Primary
2h	Standby	Primary
3h	Unavailable	Primary
4h to Dh	Reserved	
Eh	Offline	Secondary
Fh	Transitioning between states	Primary

If any of the T\_SUP bit, O\_SUP bit, U\_SUP bit, S\_SUP bit, AN\_SUP bit, or AO\_SUP bit are set to one, then the T\_SUP bit, O\_SUP bit, U\_SUP bit, S\_SUP bit, AN\_SUP bit, and AO\_SUP bit are as defined in this standard. If the T\_SUP bit, O\_SUP bit, U\_SUP bit, S\_SUP bit, AN\_SUP bit, and AO\_SUP bit are all set to zero, then which target port asymmetric access states are supported is vendor specific.

A transitioning supported (T\_SUP) bit set to one indicates that the device server supports returning the ASYMMETRIC ACCESS STATE field set to Fh (i.e., transitioning between states). A T\_SUP bit set to zero indicates that the device server does not return an ASYMMETRIC ACCESS STATE field set to Fh.

An offline supported (O\_SUP) bit set to one indicates that the offline secondary target port asymmetric access state is supported. A O\_SUP bit set to zero indicates that the offline secondary target port asymmetric access state is not supported.

An unavailable supported (U\_SUP) bit set to one indicates that the unavailable primary target port asymmetric access state is supported. A U\_SUP bit set to zero indicates that the unavailable primary target port asymmetric access state is not supported.

A standby supported (S\_SUP) bit set to one indicates that the standby primary target port asymmetric access state is supported. An S\_SUP bit set to zero indicates that the standby primary target port asymmetric access state is not supported.

An active/non-optimized supported (AN\_SUP) bit set to one indicates that the active/non-optimized primary target port asymmetric access state is supported. An AN\_SUP bit set to zero indicates that the active/non-optimized primary target port asymmetric access state is not supported.

An active/optimized supported (AO\_SUP) bit set to one indicates that the active/optimized primary target port asymmetric access state is supported. An AO\_SUP bit set to zero indicates that the active/optimized primary target port asymmetric access state is not supported.

The TARGET PORT GROUP field contains an identification of the target port group described by this target port group descriptor. Target port group information is also returned in the Device Identification VPD page (see 7.7.3).

The STATUS CODE field (see table 245) indicates why a target port group may be in a specific target port asymmetric access state. It provides a mechanism to indicate error conditions.

**Table 245 — STATUS CODE field**

Code	Description
00h	No status available.
01h	The target port asymmetric access state altered by SET TARGET PORT GROUPS command.
02h	The target port asymmetric access state altered by implicit asymmetrical logical unit access behavior.
03h to FFh	Reserved

The TARGET PORT COUNT field indicates the number of target ports that are in that target port group and the number of target port descriptors in the target port group descriptor. Every target port group shall contain at least one target port. The target port group descriptor shall include one target port descriptor for each target port in the target port group.

**Table 246 — Target port descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	Obsolete							
1								
2	(MSB)	RELATIVE TARGET PORT IDENTIFIER						
3								(LSB)

The RELATIVE TARGET PORT IDENTIFIER field contains a relative port identifier (see 3.1.120) of a target port in the target port group.

## 6.28 REPORT TIMESTAMP command

The REPORT TIMESTAMP command (see table 247) requests that the device server return the value of the logical unit's timestamp.

The REPORT TIMESTAMP command is a service action of the MAINTENANCE IN command. Additional MAINTENANCE IN service actions are defined in SCC-2 and in this standard. The MAINTENANCE IN service actions defined only in SCC-2 apply only to logical units that return a device type of 0Ch or the sccs bit set to one in their standard INQUIRY data (see 6.4.2).

**Table 247 — REPORT TIMESTAMP command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A3h)							
1	Reserved			SERVICE ACTION (0Fh)				
2	Reserved							
5								
6	(MSB)	ALLOCATION LENGTH						(LSB)
9								
10	Reserved							
11	CONTROL							

The ALLOCATION LENGTH field is defined in 4.3.5.6.

The format for the parameter data returned by the REPORT TIMESTAMP command is shown in table 248.

**Table 248 — REPORT TIMESTAMP parameter data format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____							
1	TIMESTAMP PARAMETER DATA LENGTH (0Ah) _____ (LSB)							
2	Reserved				TIMESTAMP ORIGIN			
3	Reserved							
4	TIMESTAMP							
9	TIMESTAMP							
10	Reserved							
11	Reserved							

The TIMESTAMP PARAMETER DATA LENGTH field indicates the number of bytes of parameter data that follow. The relationship between the TIMESTAMP PARAMETER DATA LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The TIMESTAMP ORIGIN field indicates the origin of the timestamp (see 5.13).

The TIMESTAMP field contains the current value of the timestamp (see 5.13).

## 6.29 REQUEST SENSE command

The REQUEST SENSE command (see table 249) requests that the device server transfer parameter data containing sense data to the application client.

**Table 249 — REQUEST SENSE command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (03h)							
1	Reserved							DESC
2	Reserved							
3								
4	ALLOCATION LENGTH							
5	CONTROL							

The descriptor format (DESC) bit (see table 250) specifies which sense data format the device server shall return in the parameter data.

**Table 250 — DESC bit**

Code	Descriptor format sense data supported?	Description
0b	yes or no	The device server shall return fixed format sense data (see 4.5.3) in the parameter data.
1b	yes	The device server shall return descriptor format sense data (see 4.5.2) in the parameter data.
	no	The device server shall return no parameter data and terminate the REQUEST SENSE command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

The ALLOCATION LENGTH field is defined in 4.3.5.6. Application clients should request 252 bytes of sense data to ensure they retrieve all the sense data. If fewer than 252 bytes are requested, sense data may be lost since the REQUEST SENSE command with any allocation length clears the sense data.

Sense data shall be available and cleared under the conditions defined in SAM-4. If the device server has no sense data available to return, it shall:

- 1) Return parameter data containing sense data with the sense key set to NO SENSE and the additional sense code set to NO ADDITIONAL SENSE INFORMATION; and
- 2) Complete the REQUEST SENSE command with GOOD status.

If the logical unit is in the idle power condition (see 5.10), the device server shall process a REQUEST SENSE command by:

- 1) Returning parameter data containing sense data with the sense key set to NO SENSE and the additional sense code set to one of the following:
  - A) LOW POWER CONDITION ON if the reason for entry into the idle power condition is unknown;

- B) IDLE CONDITION ACTIVATED BY TIMER if the logical unit entered the idle power condition due to the idle condition timer (see 7.4.12); and
  - C) IDLE CONDITION ACTIVATED BY COMMAND if the logical unit entered the idle power condition due to receipt of a command requiring the idle power condition while it was in the standby power condition; and
- 2) Complete the REQUEST SENSE command with GOOD status.

If the logical unit is in the standby power condition, the device server shall process a REQUEST SENSE command by:

- 1) Return parameter data containing sense data with the sense key set to NO SENSE and the additional sense code set to one of the following:
  - A) LOW POWER CONDITION ON if the reason for entry into the standby power condition is unknown; and
  - B) STANDBY CONDITION ACTIVATED BY TIMER if the logical unit entered the standby power condition due to the standby condition timer (see 7.4.12); and
- 2) Complete the REQUEST SENSE command with GOOD status.

Upon completion of the REQUEST SENSE command, the logical unit shall return to the same power condition that was active before the REQUEST SENSE command was received. A REQUEST SENSE command shall not reset any power condition timers.

The device server shall return CHECK CONDITION status for a REQUEST SENSE command only to report exception conditions specific to the REQUEST SENSE command itself. Examples of conditions that cause a REQUEST SENSE command to return CHECK CONDITION status are:

- a) An invalid field value is detected in the CDB;
- b) The device server does not support the REQUEST SENSE command (see 4.3.1);
- c) An unrecovered error is detected by a SCSI target port; or
- d) A malfunction prevents return of the sense data.

If a device server terminated a REQUEST SENSE command with CHECK CONDITION status, any parameter data that was transferred is invalid (i.e., the parameter data does not contain sense data).

If a REQUEST SENSE command is received on an L\_T nexus with a pending unit attention condition (i.e., before the device server reports CHECK CONDITION status) and there is an exception condition specific to the REQUEST SENSE command itself, then the device server shall not clear the pending unit attention condition (see SAM-4).

If a recovered error occurs during the processing of the REQUEST SENSE command, the device server shall:

- 1) Return parameter data containing sense data with the sense key set to RECOVERED ERROR; and
- 2) Complete the REQUEST SENSE command with GOOD status.

In response to a REQUEST SENSE command issued to a logical unit that reports a peripheral qualifier of 011b in its standard INQUIRY data (see 6.4.2) the device server shall:

- 1) Return parameter data containing sense data with the sense key set to ILLEGAL REQUEST and the additional sense code shall be set to LOGICAL UNIT NOT SUPPORTED; and
- 2) Complete the REQUEST SENSE command with GOOD status.

In response to a REQUEST SENSE command issued to a logical unit that reports a peripheral qualifier of 001b in its standard INQUIRY data, the device server shall:

- 1) Return parameter data containing sense data with the sense key set to ILLEGAL REQUEST and the additional sense code shall be set to LOGICAL UNIT NOT SUPPORTED; and
- 2) Complete the REQUEST SENSE command with GOOD status.

In response to a REQUEST SENSE command issued to a logical unit that reports a peripheral qualifier of 000b in its standard INQUIRY data because it has a peripheral device connected but is not ready for access, the device server shall:

- 1) Return parameter data containing sense data appropriate to the condition that is making the logical unit not operational; and
- 2) Complete the REQUEST SENSE command with GOOD status.

In response to a REQUEST SENSE command issued to a logical unit that reports a peripheral qualifier of 000b in its standard INQUIRY data because the device server is unable to determine whether or not a peripheral device is connected, the device server shall:

- 1) Return parameter data containing sense data with the sense key set to NO SENSE and the additional sense code set to NO ADDITIONAL SENSE INFORMATION; and
- 2) Complete the REQUEST SENSE command with GOOD status.

Device servers shall return at least 18 bytes of parameter data in response to a REQUEST SENSE command if the allocation length is 18 or greater and the DESC bit is set to zero. Application clients may determine how much sense data has been returned by examining the ALLOCATION LENGTH field in the CDB and the ADDITIONAL SENSE LENGTH field in the sense data. Device servers shall not adjust the additional sense length to reflect truncation if the allocation length is less than the sense data available.



### 6.30 SECURITY PROTOCOL IN command

The SECURITY PROTOCOL IN command (see table 251) is used to retrieve security protocol information (see 7.6.1) or the results of one or more SECURITY PROTOCOL OUT commands (see 6.31).

**Table 251 — SECURITY PROTOCOL IN command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A2h)							
1	SECURITY PROTOCOL							
2	SECURITY PROTOCOL SPECIFIC							
3								
4	INC_512	Reserved						
5	Reserved							
6	(MSB)	ALLOCATION LENGTH						
9								
10	Reserved							
11	CONTROL							

The SECURITY PROTOCOL field (see table 252) specifies which security protocol is being used.

**Table 252 — SECURITY PROTOCOL field in SECURITY PROTOCOL IN command**

Code	Description	Reference
00h	Security protocol information	7.6.1
01h to 06h	Defined by the TCG	3.1.176
07h	CbCS	7.6.4
08h to 1Fh	Reserved	
20h	Tape Data Encryption	SSC-3
21h	Data Encryption Configuration	ADC-3
22h to 3Fh	Reserved	
40h	SA Creation Capabilities	7.6.2
41h	IKEv2-SCSI	7.6.3
42h to ECh	Reserved	
EDh	SD Card TrustedFlash specification	3.1.145
EEh	Authentication in Host Attachments of Transient Storage Devices	IEEE 1667
EFh	ATA Device Server Password Security	SAT-2
F0h to FFh	Vendor Specific	

---

Editors Note 1 - ROW: SECURITY PROTOCOL field code values 22h to 2Fh are tentatively reserved for SSC-x uses.

---

The contents of the SECURITY PROTOCOL SPECIFIC field depend on the protocol specified by the SECURITY PROTOCOL field (see table 252).

A 512 increment (INC\_512) bit set to one specifies that the ALLOCATION LENGTH field (see 4.3.5.6) expresses the maximum number of bytes available to receive data in increments of 512 bytes (e.g., a value of one means 512 bytes, two means 1 024 bytes, etc.). Pad bytes may or may not be appended to meet this length. Pad bytes shall have a value of 00h. An INC\_512 bit set to zero specifies that the ALLOCATION LENGTH field expresses the number of bytes to be transferred.

Indications of data overrun or underrun and the mechanism, if any, for processing retries depend on the protocol specified by the SECURITY PROTOCOL field (see table 252).

Any association between a previous SECURITY PROTOCOL OUT command and the data transferred by a SECURITY PROTOCOL IN command depends on the protocol specified by the SECURITY PROTOCOL field (see table 252). If the device server has no data to transfer (e.g., the results for any previous SECURITY PROTOCOL OUT commands are not yet available), the device server may transfer data indicating it has no other data to transfer.

The format of the data transferred depends on the protocol specified by the SECURITY PROTOCOL field (see table 252).

The device server shall retain data resulting from a SECURITY PROTOCOL OUT command, if any, until one of the following events is processed:

- a) Transfer of the data via a SECURITY PROTOCOL IN command from the same I\_T\_L nexus as defined by the protocol specified by the SECURITY PROTOCOL field (see table 252);
- b) Logical unit reset (See SAM-4); or
- c) I\_T nexus loss (See SAM-4) associated with the I\_T nexus that sent the SECURITY PROTOCOL OUT command.

If the data is lost due to one of these events the application client may send a new SECURITY PROTOCOL OUT command to retry the operation.

## 6.31 SECURITY PROTOCOL OUT command

The SECURITY PROTOCOL OUT command (see table 253) is used to send data to the logical unit. The data sent specifies one or more operations to be performed by the logical unit. The format and function of the operations depends on the contents of the SECURITY PROTOCOL field (see table 254). Depending on the protocol specified by the SECURITY PROTOCOL field, the application client may use the SECURITY PROTOCOL IN command (see 6.30) to retrieve data derived from these operations.

**Table 253 — SECURITY PROTOCOL OUT command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (B5h)							
1	SECURITY PROTOCOL							
2	SECURITY PROTOCOL SPECIFIC							
3								
4	INC_512	Reserved						
5	Reserved							
6	(MSB)	TRANSFER LENGTH						
9								
10	Reserved							
11	CONTROL							

The SECURITY PROTOCOL field (see table 254) specifies which security protocol is being used.

**Table 254 — SECURITY PROTOCOL field in SECURITY PROTOCOL OUT command**

Code	Description	Reference
00h	Reserved	
01h to 06h	Defined by the TCG	3.1.176
07h	CbCS	7.6.4
08h to 1Fh	Reserved	
20h	Tape Data Encryption	SSC-3
21h	Data Encryption Configuration	ADC-3
22h to 40h	Reserved	
41h	IKEv2-SCSI	7.6.3
42h to ECh	Reserved	
EDh	SD Card TrustedFlash specification	3.1.145
EEh	Authentication in Host Attachments of Transient Storage Devices	IEEE 1667
EFh	ATA Device Server Password Security	SAT-2
F0h to FFh	Vendor Specific	

---

Editors Note 2 - ROW: SECURITY PROTOCOL field code values 22h to 2Fh are tentatively reserved for

---

SSC-x uses.

---

The contents of the SECURITY PROTOCOL SPECIFIC field depend on the protocol specified by the SECURITY PROTOCOL field (see table 254).

A 512 increment (INC\_512) bit set to one specifies that the TRANSFER LENGTH field (see 4.3.5.4) expresses the number of bytes to be transferred in increments of 512 bytes (e.g., a value of one means 512 bytes, two means 1 024 bytes, etc.). Pad bytes shall be appended as needed to meet this requirement. Pad bytes shall have a value of 00h. A INC\_512 bit set to zero specifies that the TRANSFER LENGTH field indicates the number of bytes to be transferred.

Any association between a SECURITY PROTOCOL OUT command and a subsequent SECURITY PROTOCOL IN command depends on the protocol specified by the SECURITY PROTOCOL field (see table 254). Each protocol shall define whether:

- a) The device server shall complete the command with GOOD status as soon as it determines the data has been correctly received. An indication that the data has been processed is obtained by sending a SECURITY PROTOCOL IN command and receiving the results in the associated data transfer; or
- b) The device server shall complete the command with GOOD status only after the data has been successfully processed and an associated SECURITY PROTOCOL IN command is not required.

The format of the data transferred depends on the protocol specified by the SECURITY PROTOCOL field (see table 254).

## 6.32 SEND DIAGNOSTIC command

The SEND DIAGNOSTIC command (see table 255) requests the device server to perform diagnostic operations on the SCSI target device, on the logical unit, or on both. Logical units that support this command shall implement, at a minimum, the default self-test feature (i.e., the SELFTEST bit equal to one and a parameter list length of zero).

**Table 255 — SEND DIAGNOSTIC command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (1Dh)							
1	SELF-TEST CODE			PF	Reserved	SELFTEST	DEVOffL	UNITOffL
2	Reserved							
3	(MSB) _____							
4	PARAMETER LIST LENGTH _____							
5	(LSB)							
5	CONTROL							

If the SELFTEST bit is set to one, the SELF-TEST CODE field shall contain 000b. If the SELFTEST bit is set to zero, the contents of SELF-TEST CODE field are specified in table 256.

**Table 256 — SELF-TEST CODE field**

Code	Name	Description
000b		This value shall be used when the SELFTEST bit is set to one, or when the SELFTEST bit is set to zero and the PF bit is set to one.
001b	Background short self-test	The device server shall start its short self-test (see 5.6.2) in the background mode (see 5.6.3.3). The PARAMETER LIST LENGTH field shall contain zero.
010b	Background extended self-test	The device server shall start its extended self-test (see 5.6.2) in the background mode (see 5.6.3.3). The PARAMETER LIST LENGTH field shall contain zero.
011b	Reserved	
100b	Abort background self-test	The device server shall abort the current self-test running in background mode. The PARAMETER LIST LENGTH field shall contain zero. This value is only valid if a previous SEND DIAGNOSTIC command specified a background self-test function and that self-test has not completed. If either of these conditions is not met, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.
101b	Foreground short self-test	The device server shall start its short self-test (see 5.6.2) in the foreground mode (see 5.6.3.2). The PARAMETER LIST LENGTH field shall contain zero.
110b	Foreground extended self-test	The device server shall start its extended self-test (see 5.6.2) in the foreground mode (see 5.6.3.2). The PARAMETER LIST LENGTH field shall contain zero.
111b	Reserved	

A page format (PF) bit set to one specifies that the SEND DIAGNOSTIC parameters and any parameters returned by a following RECEIVE DIAGNOSTIC RESULTS command with the PCV bit set to zero shall contain a single diagnostic page as defined in 7.1.

NOTE 40 - Logical units compliant with previous versions of this standard (e.g., SPC-2) may transfer more than one diagnostic page in the SEND DIAGNOSTIC command's parameter list and by doing so may request that more than one diagnostic page be transmitted in the RECEIVE DIAGNOSTIC RESULTS command's parameter data.

A PF bit set to zero specifies that all SEND DIAGNOSTIC parameters are vendor specific. If the PARAMETER LIST LENGTH field is set to zero and the SEND DIAGNOSTIC command is not going to be followed by a corresponding RECEIVE DIAGNOSTIC RESULTS command with the PCV bit set to zero, then the application client shall set the PF bit to zero. The implementation of the PF bit is optional.

A self-test (SELFTEST) bit set to one specifies that the device server shall perform the logical unit default self-test. If the self-test successfully passes, the command shall be completed with GOOD status. If the self-test fails, the command shall be terminated with CHECK CONDITION status, with the sense key set to HARDWARE ERROR.

A SELFTEST bit set to zero specifies that the device server shall perform the diagnostic operation specified by the SELF-TEST CODE field or in the parameter list. The diagnostic operation may require the device server to return parameter data that contains diagnostic results. If the return of parameter data is not required, the return of GOOD status indicates successful completion of the diagnostic operation. If the return of parameter data is required, the device server shall either:

- a) Perform the requested diagnostic operation, prepare the parameter data to be returned and indicate completion by returning GOOD status. The application client issues a RECEIVE DIAGNOSTIC RESULTS command to recover the parameter data; or
- b) Accept the parameter list, and if no errors are detected in the parameter list, return GOOD status. The requested diagnostic operation and the preparation of the parameter data to be returned are performed upon receipt of a RECEIVE DIAGNOSTIC RESULTS command.

A unit offline (UNITOFFL) bit set to one specifies that the device server may perform diagnostic operations that may affect the user accessible medium on the logical unit (e.g., write operations to the user accessible medium, or repositioning of the medium on sequential access devices). The device server may ignore the UNITOFFL bit. A UNITOFFL bit set to zero prohibits any diagnostic operations that may be detected by subsequent tasks. When the SELFTEST bit is set to zero, the UNITOFFL bit shall be ignored.

A SCSI target device offline (DEVOFFL) bit set to one grants permission to the device server to perform diagnostic operations that may affect all the logical units in the SCSI target device (e.g., alteration of reservations, log parameters, or sense data). The device server may ignore the DEVOFFL bit. A DEVOFFL bit set to zero prohibits diagnostic operations that may be detected by subsequent tasks. When the SELFTEST bit is set to zero, the DEVOFFL bit shall be ignored.

The PARAMETER LIST LENGTH field specifies the length in bytes of the parameter list that shall be transferred from the application client Data-Out Buffer to the device server. A parameter list length of zero specifies that no data shall be transferred. This condition shall not be considered an error. If PF bit is set to one and the specified parameter list length results in the truncation of the diagnostic page (e.g., the parameter list length does not match the page length specified in the diagnostic page), then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

NOTE 41 - To ensure that the diagnostic command information is not destroyed by a command sent from another I\_T nexus, the logical unit should be reserved.

### 6.33 SET IDENTIFYING INFORMATION command

The SET IDENTIFYING INFORMATION command (see table 257) requests that the device server set identifying information (see 5.15) in the logical unit to the value received in the SET IDENTIFYING INFORMATION parameter list. The SET IDENTIFYING INFORMATION command is an extension to the SET PERIPHERAL DEVICE/COMPONENT DEVICE IDENTIFIER service action of the MAINTENANCE OUT command defined in SCC-2. Additional MAINTENANCE IN and MAINTENANCE OUT service actions are defined in SCC-2 and in this standard.

The MAINTENANCE OUT service actions defined only in SCC-2 shall apply only to logical units that return a device type of 0Ch (i.e., storage array controller device) or the sccs bit set to one in their standard INQUIRY data (see 6.4.2). When a logical unit returns a device type of 0Ch or the sccs bit set to one in its standard INQUIRY data, the implementation requirements for the SCC-2 MAINTENANCE OUT service actions shall be as specified in SCC-2. Otherwise the MAINTENANCE OUT service action definitions and implementation requirements stated in this standard shall apply.

Processing a SET DEVICE IDENTIFYING INFORMATION command may require the enabling of a nonvolatile memory within the logical unit. If the nonvolatile memory is not ready, the command shall be terminated with CHECK CONDITION status, and not wait for the nonvolatile memory to become ready. The sense key shall be set to NOT READY and the additional sense code shall be set as described in table 270 (see 6.37). This information should allow the application client to determine the action required to cause the device server to become ready.

On successful completion of a SET IDENTIFYING INFORMATION command that changes identifying information saved by the logical unit, the device server shall establish a unit attention condition (see SAM-4) for the initiator port associated with every I\_T nexus except the I\_T nexus on which the SET IDENTIFIER command was received, with the additional sense code set to DEVICE IDENTIFIER CHANGED.

**Table 257 — SET IDENTIFYING INFORMATION command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A4h)							
1	Reserved			SERVICE ACTION (06h)				
2	Reserved							
3								
4	Restricted (see SCC-2)							
5								
6	(MSB) _____ PARAMETER LIST LENGTH _____ (LSB)							
9								
10	INFORMATION TYPE							Reserved
11	CONTROL							

The PARAMETER LIST LENGTH field specifies the length in bytes of the identifying information that shall be transferred from the application client to the device server. A parameter list length of zero specifies that no data shall be transferred, and that subsequent REPORT IDENTIFYING INFORMATION commands shall return the INFORMATION LENGTH field set to zero for the specified information type.

The INFORMATION TYPE field (see table 258) specifies the identifying information type to be set.

**Table 258 — INFORMATION TYPE field**

Code	Description
0000000b	Peripheral device identifying information (see 5.15).  If the PARAMETER LIST LENGTH field is set to greater than the maximum length of the peripheral device identifying information (see 5.15), the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.
0000010b	Peripheral device text identifying information (see 5.15).  If the PARAMETER LIST LENGTH field is set to a value greater than the maximum length of the peripheral device text identifying information (see 5.15), the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.  If the format of the INFORMATION field is incorrect, the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.
xxxxxx1b	Restricted (see SCC-2)
All other	Reserved

The SET IDENTIFYING INFORMATION parameter list (see table 259) contains the identifying information to be set by the device server.

**Table 259 — SET IDENTIFYING INFORMATION parameter list**

Bit Byte	7	6	5	4	3	2	1	0
0	INFORMATION							
n								

The INFORMATION field specifies the identifying information to be set for the specified information type (see 5.15).

Upon successful completion of a SET IDENTIFYING INFORMATION command, the identifying information that is saved by the logical unit shall persist through logical unit resets, hard resets, power loss, I\_T nexus losses, media format operations, and media replacement.



## 6.34 SET PRIORITY command

The SET PRIORITY command (see table 260) requests that a priority be set to the specified value. The priority set by this command shall remain in effect until one of the following occurs:

- a) Another SET PRIORITY command is received;
- b) Hard reset;
- c) Logical unit reset; or
- d) Power off.

The priority set by this command shall not be affected by an I\_T nexus loss.

The priority set by a SET PRIORITY command may be used as a command priority (see SAM-4) for tasks received by the logical unit via an I\_T nexus (i.e., an I\_T\_L nexus).

The SET PRIORITY command is a service action of the MAINTENANCE OUT command. Additional MAINTENANCE OUT service actions are defined in SCC-2 and in this standard. The MAINTENANCE OUT service actions defined only in SCC-2 apply only to logical units that return a device type of 0Ch or the sccs bit set to one in their standard INQUIRY data.

**Table 260 — SET PRIORITY command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A4h)							
1	Reserved			SERVICE ACTION (0Eh)				
2	I_T_L NEXUS TO SET		Reserved					
3	Reserved							
5								
6	(MSB)							
9	PARAMETER LIST LENGTH							
	(LSB)							
10	Reserved							
11	CONTROL							

The I\_T\_L NEXUS TO SET field (see table 261) specifies the I\_T\_L nexus and the location of the priority value to be assigned to that I\_T\_L nexus.

**Table 261 — I\_T\_L NEXUS TO SET field**

Code	Description
00b	The priority for the I_T_L nexus associated with this command shall be set to the value contained in the PRIORITY TO SET field in the SET PRIORITY parameter list (see table 262). All fields in the SET PRIORITY parameter list except the PRIORITY TO SET field shall be ignored. If the parameter list length is zero, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.
01b	The priority for the I_T_L nexus specified by the logical unit that is processing this command, the RELATIVE TARGET PORT IDENTIFIER field, and the TRANSPORTID field in the SET PRIORITY parameter list (see table 262) shall be set to the value specified by the PRIORITY TO SET field in the SET PRIORITY parameter list. If the parameter list length results in the truncation of the RELATIVE TARGET PORT IDENTIFIER field, the ADDITIONAL DESCRIPTOR LENGTH field, or the TRANSPORTID field, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR. On successful completion of a SET PRIORITY command the device server shall establish a unit attention condition (see SAM-4) for the initiator port associated with the I_T nexus specified by the TRANSPORTID field and the RELATIVE TARGET PORT IDENTIFIER field, with the additional sense code set to PRIORITY CHANGED.
10b	The priority value specified in the INITIAL COMMAND PRIORITY field of the Control Extension mode page (see 7.4.7) shall be used for all I_T_L nexuses associated with the logical unit that is processing this command regardless of any prior priority. The contents of the SET PRIORITY parameter list shall be ignored. On successful completion of a SET PRIORITY command the device server shall establish a unit attention condition (see SAM-4) for the initiator port associated with every other I_T_L nexus, with the additional sense code set to PRIORITY CHANGED.
11b	Reserved

The PARAMETER LIST LENGTH field specifies the length in bytes of the SET PRIORITY parameter list (see table 262) that shall be contained in the Data-Out Buffer. A parameter list length of zero specifies that the Data-Out Buffer shall be empty. This condition shall not be considered as an error.

**Table 262 — SET PRIORITY parameter list format**

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved				PRIORITY TO SET			
1	Reserved							
2	(MSB) _____							
3	RELATIVE TARGET PORT IDENTIFIER _____ (LSB)							
4	Reserved							
5	Reserved							
6	(MSB) _____							
7	ADDITIONAL LENGTH (n-7) _____ (LSB)							
8	_____							
n	TRANSPORTID _____							

The PRIORITY TO SET field specifies the priority to be assigned to the I\_T\_L nexus specified by the I\_T\_L NEXUS TO SET field in the CDB. The value in the PRIORITY TO SET field shall be returned in subsequent REPORT PRIORITY commands (see 6.24) until one of the conditions described in this subclause occurs. A priority to set value of zero specifies the I\_T\_L nexus specified by the I\_T\_L NEXUS TO SET field shall be set to the value specified in the INITIAL COMMAND PRIORITY field of the Control Extension mode page (see 7.4.7). The contents of the I\_T\_L NEXUS TO SET field may specify that the PRIORITY TO SET field shall be ignored.

The RELATIVE TARGET PORT IDENTIFIER field contains the relative port identifier (see 3.1.120) of the target port that is part of the I\_T\_L nexus for which the priority is to be set. The contents of the I\_T\_L NEXUS TO SET field may specify that the RELATIVE TARGET PORT IDENTIFIER field shall be ignored.

The ADDITIONAL LENGTH field specifies the number of bytes that follow in the SET PRIORITY parameter list (i.e., the size of the TransportID).

The TRANSPORTID field contains a TransportID (see 7.5.4) identifying the initiator port that is part of the I\_T\_L nexus for which the priority is to be set. The contents of the I\_T\_L NEXUS TO SET field may specify that the TRANSPORTID field shall be ignored.

### 6.35 SET TARGET PORT GROUPS command

The SET TARGET PORT GROUPS command (see table 263) requests the device server to set the primary target port asymmetric access state of all of the target ports in the specified primary target port groups and/or the secondary target port asymmetric access state of the specified target ports. See 5.9 for details regarding the transition between target port asymmetric access states. This command is mandatory for all logical units that report in the standard INQUIRY data (see 6.4.2) that they support explicit asymmetric logical units access (i.e., the TPGS field contains either 10b or 11b).

The SET TARGET PORT GROUPS command is a service action of the MAINTENANCE OUT command. Additional MAINTENANCE OUT service actions are defined in SCC-2 and in this standard. The MAINTENANCE OUT service actions defined only in SCC-2 apply only to logical units that return a device type of 0Ch or the sccs bit set to one in their standard INQUIRY data.

**Table 263 — SET TARGET PORT GROUPS command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A4h)							
1	Reserved			SERVICE ACTION (0Ah)				
2	Reserved							
5								
6	(MSB)							
9	PARAMETER LIST LENGTH						(LSB)	
10	Reserved							
11	CONTROL							

The PARAMETER LIST LENGTH field specifies the length in bytes of the target port group management parameters that shall be transferred from the application client to the device server. A parameter list length of zero specifies that no data shall be transferred, and that no change shall be made in the target port asymmetric access state of any target port groups or target ports. If the parameter list length violates the vendor specific length requirements, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The allowable values to which target port asymmetric access states may be set is vendor specific and should be reported in the REPORT TARGET PORT GROUP parameter data (see 6.27).

Primary target port groups that are not specified in a parameter list may change primary target port asymmetric access states as a result of the SET TARGET PORT GROUPS command. This shall not be considered an implicit target port asymmetric access state change.

If the SET TARGET PORT GROUPS attempts to establish an invalid combination of target port asymmetric access states or attempts to establish an unsupported target port asymmetric access state, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the SET TARGET PORT GROUPS command has been performed, the completion of the command depends upon which of the following conditions apply:

- a) If the transition is treated as a single indivisible event (see 5.9.2.5), then the SET TARGET PORT GROUPS command shall not complete until the transition to the requested state has completed; or
- b) If the transition is not treated as a single indivisible event (i.e., the device server supports other commands (see 5.9.2.5) when those commands are routed through a target port that is transitioning between target port asymmetric access states), then the SET TARGET PORT GROUPS command may complete before the transition into the requested state has completed.

If the SET TARGET PORT GROUPS command is not performed successfully, the completion of the command depends upon which of the following conditions apply:

- a) If the processing of a SET TARGET PORT GROUPS command requires the enabling of a nonvolatile memory and the nonvolatile memory is not ready, then the command shall be terminated with CHECK CONDITION status, rather than wait for the logical unit to become ready. The sense key shall be set to NOT READY and the additional sense code shall be set as described in table 270 (see 6.37); or
- b) If a failure occurred before the transition was completed, the command shall be terminated with CHECK CONDITION status, with the sense key set to HARDWARE ERROR, and the additional sense code set to SET TARGET PORT GROUPS COMMAND FAILED.

If two SET TARGET PORT GROUPS commands are performed concurrently, the target port asymmetric access state change behavior is vendor specific. A SCSI target device should not process multiple SET TARGET PORT GROUPS concurrently.

The SET TARGET PORT GROUPS parameter data format is shown in table 264.

**Table 264 — SET TARGET PORT GROUPS parameter list format**

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved							
3								
	Set target port group descriptor(s)							
4	Set target port group descriptor 0 (see table 265)							
7								
	⋮							
n-3	Set target port group descriptor x (see table 265)							
n								

The format of the set target port group descriptor is defined in table 265.

**Table 265 — Set target port group descriptor parameter list**

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved				ASYMMETRIC ACCESS STATE			
1	Reserved							
2	(MSB) _____							
3	_____ (LSB)							

If the ASYMMETRIC ACCESS STATE field (see table 266) specifies a primary target port asymmetric access state, then all the target ports in the specified target port group shall transition to the specified state (see 5.9.2.5). If the ASYMMETRIC ACCESS STATE field specifies a secondary target port asymmetric access state, then the specified target port shall transition to the specified state.

**Table 266 — ASYMMETRIC ACCESS STATE field**

Value	State (see 5.9.2.4)	Type (see 5.9.2.1)
0h	Active/optimized	Primary
1h	Active/non-optimized	Primary
2h	Standby	Primary
3h	Unavailable	Primary
4h to Dh	Reserved	
Eh	Offline	Secondary
Fh	Illegal Request <sup>a</sup>	
<sup>a</sup> If the ASYMMETRIC ACCESS STATE field in any target port group descriptor contains Fh, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.		

If the ASYMMETRIC ACCESS STATE field (see table 266) specifies a primary target port asymmetric access state, then the TARGET PORT GROUP OR TARGET PORT field specifies a primary target port group for which the primary target port asymmetric access state shall be changed. If the ASYMMETRIC ACCESS STATE field specifies a secondary target port asymmetric access state, then the TARGET PORT GROUP OR TARGET PORT field specifies the relative target port identifier (see 3.1.120) of the target port for which the secondary target port asymmetric access state shall be changed.

### 6.36 SET TIMESTAMP command

The SET TIMESTAMP command (see table 267) requests the device server to initialize the timestamp (see 5.13), if the SCSIP bit is set to one or the TCMOS bit is set to one in the Control Extension mode page (see 7.4.7). If the SCSIP bit is set to zero, the SET TIMESTAMP command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The SET TIMESTAMP command is a service action of the MAINTENANCE OUT command. Additional MAINTENANCE OUT service actions are defined in SCC-2 and in this standard. The MAINTENANCE OUT service actions defined only in SCC-2 apply only to logical units that return a device type of 0Ch or the sccs bit set to one in their standard INQUIRY data (see 6.4.2).

**Table 267 — SET TIMESTAMP command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A4h)							
1	Reserved			SERVICE ACTION (0Fh)				
2	Reserved							
5								
6	(MSB)	PARAMETER LIST LENGTH						(LSB)
9								
10	Reserved							
11	CONTROL							

The PARAMETER LIST LENGTH field specifies the length in bytes of the SET TIMESTAMP parameters that shall be transferred from the application client to the device server. A parameter list length of zero indicates that no data shall be transferred, and that no change shall be made to the timestamp.

The format for the parameter data returned by the SET TIMESTAMP command is shown in table 268.

**Table 268 — SET TIMESTAMP parameter data format**

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved							
3								
4	TIMESTAMP							
9								
10	Reserved							
11	Reserved							

The TIMESTAMP field shall contain the initial value of the timestamp in the format defined in 5.13. The timestamp should be the number of milliseconds that have elapsed since midnight, 1 January 1970 UT. If the high order byte in the TIMESTAMP field is greater than F0h, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

On successful completion of a SET TIMESTAMP command the device server shall establish a unit attention condition for the initiator port associated with every I\_T nexus except the I\_T nexus on which the SET TIMESTAMP command was received (see SAM-4), with the additional sense code set to TIMESTAMP CHANGED.

### 6.37 TEST UNIT READY command

The TEST UNIT READY command (see table 269) provides a means to check if the logical unit is ready. This is not a request for a self-test. If the logical unit is able to accept an appropriate medium-access command without returning CHECK CONDITION status, this command shall return a GOOD status. If the logical unit is unable to become operational or is in a state such that an application client action (e.g., START UNIT command) is required to make the logical unit ready, the command shall be terminated with CHECK CONDITION status, with the sense key set to NOT READY.

**Table 269 — TEST UNIT READY command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (00h)							
1	Reserved							
4								
5	CONTROL							

Table 270 defines the suggested GOOD and CHECK CONDITION status responses to the TEST UNIT READY command. Other conditions, including deferred errors, may result in other responses (e.g., BUSY status or RESERVATION CONFLICT status).

**Table 270 — Preferred TEST UNIT READY responses**

Status	Sense Key	Additional Sense Code
GOOD	not applicable	not applicable
CHECK CONDITION	ILLEGAL REQUEST	LOGICAL UNIT NOT SUPPORTED
CHECK CONDITION	NOT READY	LOGICAL UNIT DOES NOT RESPOND TO SELECTION
CHECK CONDITION	NOT READY	MEDIUM NOT PRESENT
CHECK CONDITION	NOT READY	LOGICAL UNIT NOT READY, CAUSE NOT REPORTABLE
CHECK CONDITION	NOT READY	LOGICAL UNIT IS IN PROCESS OF BECOMING READY
CHECK CONDITION	NOT READY	LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED
CHECK CONDITION	NOT READY	LOGICAL UNIT NOT READY, MANUAL INTERVENTION REQUIRED
CHECK CONDITION	NOT READY	LOGICAL UNIT NOT READY, FORMAT IN PROGRESS



## 6.38 WRITE ATTRIBUTE command

The WRITE ATTRIBUTE command (see table 271) allows an application client to write attributes to medium auxiliary memory. Device servers that implement the WRITE ATTRIBUTE command shall also implement the READ ATTRIBUTE command (see 6.15). Application clients should issue READ ATTRIBUTE commands prior to using this command to discover device server support for medium auxiliary memory.

**Table 271 — WRITE ATTRIBUTE command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (8Dh)							
1	Reserved							
2	Restricted (see SMC-2)							
4								
5	VOLUME NUMBER							
6	Reserved							
7	PARTITION NUMBER							
8	Reserved							
9								
10	(MSB)	PARAMETER LIST LENGTH						(LSB)
13								
14	Reserved							
15	CONTROL							

The VOLUME NUMBER field specifies a volume (see SSC-2) within the medium auxiliary memory. The number of volumes of the medium auxiliary memory shall equal that of the attached medium. If the medium only has a single volume, then its volume number shall be zero.

The PARTITION NUMBER field specifies a partition (see SSC-2) within a volume. The number of partitions of the medium auxiliary memory shall equal that of the attached medium. If the medium only has a single partition, then its partition number shall be zero.

If the combination of volume number and partition number is not valid, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The PARAMETER LIST LENGTH field specifies the length in bytes of the parameter list contained in the Data-Out Buffer. A parameter list length of zero specifies that no parameter data is present; this shall not be considered an error. If the parameter list length results in the truncation of an attribute, the WRITE ATTRIBUTE command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

The parameter list shall have the format shown in table 272. Attributes should be sent in ascending numerical order. If the attributes are not in order, then no attributes shall be changed and the WRITE ATTRIBUTE command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

Table 272 — WRITE ATTRIBUTE parameter list format

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
3	PARAMETER DATA LENGTH (n-3) (LSB)							
	Attribute(s)							
4	Attribute 0 (see 7.3.1)							
	⋮							
n	Attribute x (see 7.3.1)							

The PARAMETER DATA LENGTH field should contain the number of bytes of attribute data and shall be ignored by the device server.

The format of the attributes is described in 7.3.1.

If there is not enough space to write the attributes to the medium auxiliary memory, then no attributes shall be changed and the WRITE ATTRIBUTE command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to AUXILIARY MEMORY OUT OF SPACE.

If the medium auxiliary memory is not accessible because there is no medium present, then no attributes shall be changed and the WRITE ATTRIBUTE command shall be terminated with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to MEDIUM NOT PRESENT.

If the medium is present but the medium auxiliary memory is not accessible, then no attributes shall be changed and the WRITE ATTRIBUTE command shall be terminated with CHECK CONDITION status, with the sense key set to MEDIUM ERROR, and the additional sense code set to LOGICAL UNIT NOT READY, AUXILIARY MEMORY NOT ACCESSIBLE.

If the medium auxiliary memory is not operational (e.g., bad checksum), the WRITE ATTRIBUTE command shall be terminated with CHECK CONDITION status, with the sense key set to MEDIUM ERROR, and the additional sense code set to AUXILIARY MEMORY WRITE ERROR.

If the WRITE ATTRIBUTE command parameter data contains an attribute with an ATTRIBUTE LENGTH field (see 7.3.1) set to zero, then one of the following actions shall occur:

- a) If the attribute state is unsupported or read only (see 5.11), then no attributes shall be changed and the WRITE ATTRIBUTE command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST;

- b) If the attribute state is read/write, the attribute shall be changed to the nonexistent state. This attribute shall not be returned in response to a READ ATTRIBUTE command and not be reported by the READ ATTRIBUTE command with ATTRIBUTE LIST service action; or
- c) If the attribute state is nonexistent, the attribute in the WRITE ATTRIBUTE command parameter list shall be ignored; this shall not be considered an error.

No attributes shall be changed, the WRITE ATTRIBUTE command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST if the parameter data contains any of the following:

- a) An attempt to change an attribute in the read only state (see 5.11);
- b) An attribute with incorrect ATTRIBUTE LENGTH field (see 7.3.1) contents; or
- c) An attribute with unsupported ATTRIBUTE VALUE field (see 7.3.1) contents.

## 6.39 WRITE BUFFER command

### 6.39.1 WRITE BUFFER command introduction

The WRITE BUFFER command (see table 273) is used in conjunction with the READ BUFFER command for:

- a) Testing logical unit buffer memory;
- b) Testing the integrity of the service delivery subsystem;
- c) Downloading microcode (see 5.16); and
- d) Downloading application client error history (see 5.12).

**Table 273 — WRITE BUFFER command**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (3Bh)							
1	Reserved			MODE				
2	BUFFER ID							
3	(MSB) _____							
5	BUFFER OFFSET _____ (LSB)							
6	(MSB) _____							
8	PARAMETER LIST LENGTH _____ (LSB)							
9	CONTROL							

This command shall not alter any medium of the logical unit when the data mode or the combined header and data mode is specified.

The function of this command and the meaning of fields within the CDB depend on the contents of the MODE field. The MODE field is defined in table 274.

**Table 274 — WRITE BUFFER MODE field**

Code	Description	Reference
00h	Combined header and data <sup>a</sup>	6.39.2
01h	Vendor specific <sup>a</sup>	6.39.3
02h	Data	6.39.4
03h	Reserved	
04h	Download microcode and activate	5.16 and 6.39.5
05h	Download microcode, save, and activate	5.16 and 6.39.6
06h	Download microcode with offsets and activate	5.16 and 6.39.7
07h	Download microcode with offsets, save, and activate	5.16 and 6.39.8
08h to 09h	Reserved	
0Ah	Write data to echo buffer	6.39.9
0Bh to 0Dh	Reserved	
0Eh	Download microcode with offsets, save, and defer active	5.16 and 6.39.10
0Fh	Activate deferred microcode	5.16 and 6.39.11
10h to 19h	Reserved	
1Ah	Enable expander communications protocol and Echo buffer	6.39.12
1Bh	Disable expander communications protocol	6.39.13
1Ch	Download application client error history	5.12 and 6.39.14
1Dh to 1Fh	Reserved	
<sup>a</sup> Modes 00h and 01h are not recommended.		

### 6.39.2 Combined header and data mode (00h)

In this mode, data to be transferred is preceded by a four-byte header. The four-byte header consists of all reserved bytes.

The BUFFER ID and the BUFFER OFFSET fields shall be set to zero.

The PARAMETER LIST LENGTH field specifies the maximum number of bytes that shall be transferred from the Data-Out Buffer. This number includes four bytes of header, so the data length to be stored in the device server's buffer is parameter list length minus four. The application client should attempt to ensure that the parameter list length is not greater than four plus the BUFFER CAPACITY field value (see 6.16.2) that is returned in the header of the READ BUFFER command combined header and data mode (i.e., 0h). If the parameter list length exceeds the buffer capacity, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

### 6.39.3 Vendor specific mode (01h)

In this mode, the meaning of the BUFFER ID, BUFFER OFFSET, and PARAMETER LIST LENGTH fields are not specified by this standard.

#### 6.39.4 Data mode (02h)

In this mode, the Data-Out Buffer contains buffer data destined for the logical unit. The BUFFER ID field identifies a specific buffer within the logical unit. The vendor assigns buffer ID codes to buffers within the logical unit. Buffer ID zero shall be supported. If more than one buffer is supported, then additional buffer ID codes shall be assigned contiguously, beginning with one. If an unsupported buffer ID code is selected, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The BUFFER OFFSET field specifies the location in the buffer to which the data is written. The application client should conform to the offset boundary requirements returned in the READ BUFFER descriptor (see 6.16.5). If the device server is unable to process the specified buffer offset, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The PARAMETER LIST LENGTH field specifies the maximum number of bytes that shall be transferred from the Data-Out Buffer to be stored in the specified buffer beginning at the buffer offset. The application client should attempt to ensure that the parameter list length plus the buffer offset does not exceed the capacity of the specified buffer. The capacity of the buffer is indicated by the BUFFER CAPACITY field in the READ BUFFER descriptor (see 6.16.5). If the BUFFER OFFSET and PARAMETER LIST LENGTH fields specify a transfer in excess of the buffer capacity, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

#### 6.39.5 Download microcode and activate mode (04h)

In this mode, microcode shall be transferred to the device server and activated (see 5.16).

The BUFFER ID field, BUFFER OFFSET field, and PARAMETER LIST LENGTH field are vendor specific.

#### 6.39.6 Download microcode, save, and activate mode (05h)

In this mode, microcode shall be transferred to the device server, saved to nonvolatile storage, and activated (see 5.16).

The downloaded microcode may or may not be activated after the WRITE BUFFER command completes and shall be activated when one of the following occurs:

- a) Power on; or
- b) After each hard reset.

The BUFFER ID field, BUFFER OFFSET field, and PARAMETER LIST LENGTH field are vendor specific.

#### 6.39.7 Download microcode with offsets and activate mode (06h)

In this mode, microcode shall be transferred to the device server using one or more WRITE BUFFER commands and activated (see 5.16).

The BUFFER ID field specifies a buffer within the logical unit. The vendor assigns buffer ID codes to buffers within the logical unit. A buffer ID value of zero shall be supported. If more than one buffer is supported, then additional buffer ID codes shall be assigned contiguously, beginning with one. If an unsupported buffer ID code is specified, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The BUFFER OFFSET field specifies the location in the buffer to which the microcode is written. The application client shall send commands that conform to the offset boundary requirements returned in the READ BUFFER descriptor (see 6.16.5). If the device server is unable to process the specified buffer offset, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The PARAMETER LIST LENGTH field specifies the maximum number of bytes that shall be present in the Data-Out Buffer to be stored in the specified buffer beginning at the buffer offset. The application client should ensure that the parameter list length plus the buffer offset does not exceed the capacity of the specified buffer. The capacity of the buffer is indicated by the BUFFER CAPACITY field in the READ BUFFER descriptor (see 6.16.5). If the BUFFER OFFSET and PARAMETER LIST LENGTH fields specify a transfer in excess of the buffer capacity, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

#### **6.39.8 Download microcode with offsets, save, and activate mode (07h)**

In this mode, microcode shall be transferred to the device server using one or more WRITE BUFFER commands and saved to nonvolatile storage (see 5.16).

The downloaded microcode may or may not be activated after the WRITE BUFFER command completes and shall be activated when one of the following occurs:

- a) Power on; or
- b) After each hard reset.

The BUFFER ID field, BUFFER OFFSET field, and PARAMETER LIST LENGTH field are defined in the download microcode with offsets mode (see 6.39.7).

#### **6.39.9 Write data to echo buffer mode (0Ah)**

In this mode the device server transfers data from the application client and stores it in an echo buffer. An echo buffer is assigned in the same manner by the device server as it would for a write operation. Data shall be sent aligned on four-byte boundaries.

The BUFFER ID and BUFFER OFFSET fields shall be ignored in this mode.

NOTE 42 - It is recommended that the logical unit assign echo buffers on a per I\_T nexus basis to limit the number of exception conditions that may occur when I\_T nexuses are present.

Upon successful completion of a WRITE BUFFER command the data shall be preserved in the echo buffer unless there is an intervening command to any logical unit in which case the data may be changed.

The PARAMETER LIST LENGTH field specifies the maximum number of bytes that shall be transferred from the Data-Out Buffer to be stored in the echo buffer. The application client should ensure that the parameter list length does not exceed the capacity of the echo buffer. The capacity of the echo buffer is indicated by the BUFFER CAPACITY field in the READ BUFFER echo buffer descriptor (see 6.16.7). If the PARAMETER LIST LENGTH field specifies a transfer in excess of the buffer capacity, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

#### **6.39.10 Download microcode with offsets, save, and defer activate mode (0Eh)**

In this mode, microcode shall be transferred to the device server using one or more WRITE BUFFER commands, saved to nonvolatile storage, and considered deferred (see 5.16).

The deferred microcode shall be activated and no longer considered deferred when one of the following occurs:

- a) A power on;
- b) A hard reset;
- c) A START STOP UNIT command is processed (see SBC-3);
- d) A FORMAT UNIT command is processed (see SBC-3); or
- e) A WRITE BUFFER command with the activate deferred microcode mode (0Fh) is processed (see 6.39.11).

The BUFFER ID field, BUFFER OFFSET field, and PARAMETER LIST LENGTH field are defined in the download microcode with offsets mode (see 6.39.7).

#### **6.39.11 Activate deferred microcode mode (0Fh)**

In this mode, deferred microcode that has been saved using the download microcode with offsets, save, and defer activate mode (see 6.39.10), if any, shall be activated and no longer considered deferred (see 5.16).

The the BUFFER ID field, the BUFFER OFFSET field, and PARAMETER LIST LENGTH field shall be ignored in this mode.

If there is no deferred microcode that has been saved using the download microcode with offsets, save, and defer activate mode, then the WRITE BUFFER command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to COMMAND SEQUENCE ERROR.

#### **6.39.12 Enable expander communications protocol and Echo buffer mode (1Ah)**

Receipt of a WRITE BUFFER command with this mode causes a communicative expander (see SPI-5) to enter the expanded communications protocol mode.

Device servers in SCSI target devices that receive a WRITE BUFFER command with this mode shall process it as if it were a WRITE BUFFER command with the write data to Echo buffer mode (see 6.39.9).

#### **6.39.13 Disable expander communications protocol mode (1Bh)**

Receipt of a WRITE BUFFER command with this mode causes a communicative expander (see SPI-5) to exit the expanded communications protocol mode and return to simple expander operation.

Device servers in SCSI target devices that receive a WRITE BUFFER command with this mode shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

#### **6.39.14 Download application client error history mode (1Ch)**

In this mode the device server transfers application client error history from the application client and stores it in the error history (see 5.12). The format of the application client error history parameter list is defined in table 275.

The BUFFER ID field and BUFFER OFFSET field shall be ignored in this mode.

Upon successful completion of a WRITE BUFFER command, the information contained in the application client error history parameter list shall be appended to the application client error history in a format determined by the logical unit.

The PARAMETER LIST LENGTH field specifies the length in bytes of the application client error history parameter list that shall be transferred from the application client to the device server. If the PARAMETER LIST LENGTH field specifies a transfer that exceeds the error history capacity, the command shall be terminated with CHECK CONDITION

status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The device server shall not return an error based on the contents of any of the field values defined in table 275 except:

- a) The CLR bit;
- b) The ERROR LOCATION LENGTH field; and
- c) the APPLICATION CLIENT ERROR HISTORY LENGTH field.

**Table 275 — Application client error history parameter list format**

Bit Byte	7	6	5	4	3	2	1	0	
0	(MSB)	T10 VENDOR IDENTIFICATION							(LSB)
7									
8	(MSB)	ERROR TYPE							(LSB)
9									
10		Reserved						CLR	
11		Reserved							
12	(MSB)	TIMESTAMP							(LSB)
17									
18		Reserved							
19									
20		Reserved			CODE SET				
21		ERROR LOCATION FORMAT							
22	(MSB)	ERROR LOCATION LENGTH (m-25)							(LSB)
23									
24	(MSB)	APPLICATION CLIENT ERROR HISTORY LENGTH (n-m)							(LSB)
25									
26	(MSB)	ERROR LOCATION							(LSB)
m									
m+1		APPLICATION CLIENT ERROR HISTORY							
n									

The T10 VENDOR IDENTIFICATION field contains eight bytes of left-aligned ASCII data (see 4.4.1) identifying the vendor providing the application client error history. The T10 vendor identification shall be one assigned by INCITS. A list of assigned T10 vendor identifications is in Annex E and on the T10 web site (<http://www.t10.org>).



The ERROR TYPE field (see table 276) specifies the error detected by the application client.

**Table 276 — ERROR TYPE field**

Code	Description
0000h	No error specified by the application client
0001h	An unknown error was detected by the application client
0002h	The application client detected corrupted data
0003h	The application client detected a permanent error
0004h	The application client detected a service response of SERVICE DELIVERY OR TARGET FAILURE (see SAM-4).
0005h to 7FFFh	Reserved
8000h to FFFFh	Vendor specific

If the CLR\_SUP bit is set to one in the error history directory parameter data (see 6.16.9.2), a CLR bit set to one specifies that the device server shall:

- a) Clear the portions of the error history that the device server allows to be cleared; and
- b) Ignore any application client error history specified in the parameter list.

If the CLR\_SUP bit is set to one in the error history directory parameter data, a CLR bit set to zero specifies that the device server shall not ignore the CLR bit.

If the CLR\_SUP bit is set to one in the error history directory parameter data, a CLR bit set to zero specifies that the device server shall:

- a) Not clear the error history; and
- b) Process all application client error history specified in the parameter list.

If the CLR\_SUP bit is set to zero in the error history directory parameter data, the device server shall ignore the CLR bit.

The TIMESTAMP field shall contain:

- a) A time based on the timestamp reported by the REPORT TIMESTAMP command, if the device server supports a device clock (see 5.13),
- b) The number of milliseconds that have elapsed since midnight, 1 January 1970 UT (see 3.1.178); or
- c) Zero, if the application client is not able to determine the UT of the log entry.

The CODE SET field contains a code set enumeration (see 4.4.3) that indicates the format of the APPLICATION CLIENT ERROR HISTORY field.

The ERROR LOCATION FORMAT field (see table 277) specifies the format of the ERROR LOCATION field.

**Table 277 — ERROR LOCATION FORMAT field**

Code	Description
00h	No error history location specified by the application client
01h	For direct-access block devices (see SBC-3 and RBC), the ERROR LOCATION field specifies the logical block address associated the specified application client error history. For other peripheral device types, this code is reserved.
02h to 7Fh	Reserved
80h to FFh	Vendor specific

The ERROR LOCATION LENGTH field specifies the length of the ERROR LOCATION field. The ERROR LOCATION LENGTH field value shall be a multiple of four. An ERROR LOCATION LENGTH field set to zero specifies that there is no error location information.

The APPLICATION CLIENT ERROR HISTORY LENGTH field specifies the length of the APPLICATION CLIENT ERROR HISTORY field. The APPLICATION CLIENT ERROR HISTORY LENGTH field value shall be a multiple of four. An APPLICATION CLIENT ERROR HISTORY field set to zero specifies that there is no vendor specific information.

The ERROR LOCATION field specifies the location at which the application client detected the error, in the format specified by the ERROR LOCATION FORMAT field.

The APPLICATION CLIENT ERROR HISTORY field contains vendor specific application client error history (see 5.12.1).

## 7 Parameters for all device types

### 7.1 Diagnostic parameters

#### 7.1.1 Diagnostic page format and page codes for all device types

This subclause describes the diagnostic page structure and the diagnostic pages that are applicable to all SCSI devices. Diagnostic pages specific to each device type are described in the command standard (see 3.1.27) that applies to that device type.

A SEND DIAGNOSTIC command with a PF bit set to one specifies that the SEND DIAGNOSTIC parameter list consists of a single diagnostic page and that the data returned by the subsequent RECEIVE DIAGNOSTIC RESULTS command that has the PCV bit set to zero shall use the diagnostic page format defined in table 278. A RECEIVE DIAGNOSTIC RESULTS command with a PCV bit set to one specifies that the device server return a diagnostic page using the format defined in table 278.

**Table 278 — Diagnostic page format**

Bit Byte	7	6	5	4	3	2	1	0
0	PAGE CODE							
1	Page code specific							
2	(MSB)							
3	PAGE LENGTH (n-3)							
4	(LSB)							
n	Diagnostic parameters							

Each diagnostic page defines a function or operation that the device server shall perform as a result of a SEND DIAGNOSTIC command or the information being returned as a result of a RECEIVE DIAGNOSTIC RESULTS command with the PCV bit equal to one. The diagnostic parameters contain data that is formatted according to the page code specified.

The PAGE CODE field (see table 279) identifies the diagnostic page.

**Table 279 — Diagnostic page codes**

Page Code	Diagnostic Page Name	Reference
00h	Supported Diagnostic Pages	7.1.3
01h to 2Fh	Defined by SES-2 for: a) Enclosure services devices (i.e., SCSI devices with the PERIPHERAL DEVICE TYPE field set to 0Dh in standard INQUIRY data); and b) SCSI devices with the ENCSERV bit set to one in standard INQUIRY data (see 6.4.2).	SES-2
30h to 3Eh	Reserved	
3Fh	Protocol Specific	7.1.2
40h to 7Fh	See specific device type for definition	
80h to FFh	Vendor specific	
Annex D contains a listing of diagnostic page codes in numeric order.		

The PAGE LENGTH field contains the length in bytes of the diagnostic parameters that follow this field. If the application client sends a SEND DIAGNOSTIC command with a parameter list containing a PAGE LENGTH field that results in the truncation of any parameter, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The diagnostic parameters are defined for each diagnostic page code. The diagnostic parameters within a diagnostic page may be defined differently in a SEND DIAGNOSTIC command than in a RECEIVE DIAGNOSTIC RESULTS command.

### 7.1.2 Protocol Specific

The Protocol Specific diagnostic page (see table 280) provides access to SCSI transport protocol specific diagnostic parameters.

**Table 280 — Protocol Specific diagnostic page**

Bit Byte	7	6	5	4	3	2	1	0
0	PAGE CODE (3Fh)							
1	Reserved				PROTOCOL IDENTIFIER			
2	(MSB)							
3	PAGE LENGTH (n-3)							
4	(LSB)							
n	SCSI transport protocol specific diagnostic parameters							

The PAGE CODE field is described in 7.1.1, and shall be set to 3Fh to indicate a Protocol Specific diagnostic page follows.

The PROTOCOL IDENTIFIER field contains one of the values shown in table 361 (see 7.5.1) to identify the SCSI transport protocol standard that defines the SCSI transport protocol specific diagnostic parameters. The SCSI transport protocol specific data is defined by the corresponding SCSI transport protocol standard.

The PAGE LENGTH field specifies the length in bytes of the following supported page list.

The SCSI transport protocol specific diagnostic parameters are defined by the SCSI transport protocol standard that corresponds to the value in the PROTOCOL IDENTIFIER field.

7.1.3 Supported Diagnostic Pages

The Supported Diagnostic Pages diagnostic page (see table 281) returns the list of diagnostic pages implemented by the device server. This diagnostic page shall be implemented if the device server implements the diagnostic page format option of the SEND DIAGNOSTIC and RECEIVE DIAGNOSTIC RESULTS commands.

Table 281 — Supported Diagnostic Pages diagnostic page

Bit Byte	7	6	5	4	3	2	1	0
0	PAGE CODE (00h)							
1	Reserved							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
4	SUPPORTED PAGE LIST							
n								

The definition of this diagnostic page for the SEND DIAGNOSTIC command includes only the first four bytes. If the PAGE LENGTH field is not zero, the device server shall terminate the SEND DIAGNOSTIC command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST. This diagnostic page instructs the device server to make available the list of all supported diagnostic pages to be returned by a subsequent RECEIVE DIAGNOSTIC RESULTS command.

The definition of this diagnostic page for the RECEIVE DIAGNOSTIC RESULTS command includes the list of diagnostic pages supported by the device server.

The PAGE CODE field is described in 7.1.1, and shall be set to 00h to indicate a Supported Diagnostics Pages diagnostic page follows.

The PAGE LENGTH field specifies the length in bytes of the following supported page list.

The SUPPORTED PAGE LIST field shall contain a list of all diagnostic page codes, one per byte, implemented by the device server in ascending order beginning with page code 00h.

## 7.2 Log parameters

### 7.2.1 Log page structure and log parameter structure for all device types

#### 7.2.1.1 Log page structure

This subclause describes the log page structure that is applicable to all SCSI devices. Log pages specific to each device type are described in the command standard (see 3.1.27) that applies to that device type. The LOG SELECT command (see 6.5) supports the ability to send zero or more log pages. The LOG SENSE command (see 6.6) returns a single log page specified by the combination of the PAGE CODE field and SUBPAGE CODE field in the CDB.

Each log page begins with a four-byte page header followed by zero or more variable-length log parameters defined for that log page. The log page format is defined in table 282.

**Table 282 — Log page format**

Bit Byte	7	6	5	4	3	2	1	0
0	DS	SPF	PAGE CODE					
1	SUBPAGE CODE							
2	(MSB) _____ PAGE LENGTH (n-3) _____ (LSB)							
3								
	Log parameter(s)							
4	Log parameter [first] (see 7.2.1.2) _____							
x+3	(Length x) _____							
	⋮							
	⋮							
n-y+1	Log parameter [last] (see 7.2.1.2) _____							
n	(Length y) _____							

For the LOG SENSE command (see 6.6), the DS bit indicates whether log parameters in this log page are saved when the SP bit is set to one in the CDB. If the DS bit is set to zero, the log parameters are saved when the SP bit is set to one. If the DS bit is set to one, the log parameters are not saved. For the LOG SELECT command (see 6.5), the disable save (DS) bit operates in conjunction with the parameter code reset (PCR) bit, the save parameters (SP) bit, the page control (PC) field, and the PARAMETER LIST LENGTH field in the CDB.

If the subpage format (SPF) bit is set to zero, then the SUBPAGE CODE field shall contain 00h. If the SPF bit is set to one, then the SUBPAGE CODE field shall contain a value between 01h and FFh.

The PAGE CODE field contains the number of the log page that is being transferred.

The SUBPAGE CODE field contains the subpage number of the log page that is being transferred.

If an application client specifies values in the PAGE CODE field and SUBPAGE CODE field for a log page that is reserved or not implemented by the logical unit, then the device server shall terminate the LOG SELECT command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the **PARAMETER LIST LENGTH** field in a **LOG SELECT CDB** contains zero, the meanings for the **PCR bit**, **SP bit**, and **PC field** are defined in 6.5.2.

If the **PARAMETER LIST LENGTH** field in a **LOG SELECT CDB** contains a non-zero value (i.e., when parameter data is being sent with the **LOG SELECT** command), table 283 defines the meaning for the combinations of values for:

- a) The **PCR bit**, the **SP bit**, and the **PC field** in the **LOG SELECT CDB**; and
- b) The **DS bit** in the **LOG SELECT** parameter data.

**Table 283 — LOG SELECT PCR bit, SP bit, and DS bit meanings when parameter list length is not zero**

<b>PCR bit</b>	<b>SP bit</b>	<b>DS bit</b>	<b>Description</b>
0b	0b	xb	The device server shall set the specified values <sup>a</sup> to the values in the parameter list and shall not save any values to non-volatile media.
0b	1b	0b	The device server shall set the specified values <sup>a</sup> to the values in the parameter list and shall process the optional saving of log parameter values as follows: <ul style="list-style-type: none"> <li>a) If default data counter values are specified (see table 140 in 6.5.1), no values shall be saved;</li> <li>b) If values other than default data counter values are specified and the device server implements saving of the specified values<sup>a</sup>, then the device server shall save the specified values<sup>a</sup> in the parameter list to non-volatile media; or</li> <li>c) If values other than default values are specified and the device server does not implement saving of one or more of the specified values<sup>a</sup>, then the device server shall terminate the command with <b>CHECK CONDITION</b> status, with the sense key set to <b>ILLEGAL REQUEST</b>, and the additional sense code set to <b>INVALID FIELD IN PARAMETER LIST</b>.</li> </ul>
0b	1b	1b	The device server shall set the specified values <sup>a</sup> to the values in the parameter list and shall not save any values in the specified log page to non-volatile media.
1b	xb	xb	The device server terminate the command with <b>CHECK CONDITION</b> status, with the sense key set to <b>ILLEGAL REQUEST</b> , and the additional sense code set to <b>INVALID FIELD IN CDB</b> .
<sup>a</sup> The specified parameters are determined by the <b>FORMAT AND LINKING</b> field contents (see table 286) in the <b>LOG SELECT</b> parameter data and by the <b>PC field</b> contents (see table 140 in 6.5.1) in the <b>LOG SELECT CDB</b> .			

The value in the **PAGE LENGTH** field is the length in bytes of the log parameters that follow. If the application client sends a log page length that results in the truncation of any parameter, the command shall be terminated with **CHECK CONDITION** status, with the sense key set to **ILLEGAL REQUEST**, and the additional sense code set to **INVALID FIELD IN PARAMETER LIST**.

### 7.2.1.2 Log parameter structure

#### 7.2.1.2.1 Introduction

Most log pages contain one or more special data structures called log parameters (see table 284). Log parameters may be data counters of a particular event(s), the conditions under which certain operations were performed, or list parameters that contain a character string or binary description of a particular event.

**Table 284 — Log parameter**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____							
1	PARAMETER CODE _____ (LSB)							
2	Parameter control byte (see 7.2.1.2.2)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (n-3)							
4	_____							
n	PARAMETER VALUE _____							

Each log parameter begins with a four-byte parameter header followed by one or more bytes of PARAMETER VALUE data.

The PARAMETER CODE field identifies the log parameter being transferred for that log page. If an application client specifies a value in the PARAMETER CODE field in the LOG SELECT command parameter data that is reserved or not implemented by the logical unit, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are collectively referred to as the parameter control byte. The bits and fields in the parameter control byte are described in 7.2.1.2.2.

The PARAMETER LENGTH field specifies the length in bytes of the PARAMETER VALUE field that follows. If the application client specifies a parameter length that results in the truncation of the PARAMETER VALUE field, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If an application client sends a log parameter that is not supported by the logical unit, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the application client sends a log parameter value that is outside the range supported by the logical unit, and rounding is implemented for that parameter, the device server may either:

- a) Round to an acceptable value and terminate the command as described in 5.4; or
- b) Terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the parameter data for one LOG SELECT command contains more than one log page and the log pages are not in ascending order by page code value then subpage code value, then the device server shall terminate the



command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the parameter data for one LOG SELECT command contains more than one log parameter in any one log page and the log parameters are not in ascending order by parameter code value, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

NOTE 43 - Application clients should send LOG SENSE commands prior to sending LOG SELECT commands to determine supported log pages and page lengths.

The SCSI target device may provide independent sets of log parameters for each logical unit or for each combination of logical units and I\_T nexuses. If the SCSI target device does not support independent sets of log parameters and any log parameters are changed that affect other I\_T nexuses, then the device server shall establish a unit attention condition (see SAM-4) for the initiator port associated with every I\_T nexus except the I\_T nexus on which the LOG SELECT command was received, with the additional sense code set to LOG PARAMETERS CHANGED.

#### **7.2.1.2.2 Parameter control byte**

##### **7.2.1.2.2.1 Introduction**

The bits and fields in the parameter control byte are described in 7.2.1.2.2.

For cumulative log parameter values, indicated by the PC field (see table 140 in 6.5.1) of the LOG SELECT command and LOG SENSE command, the disable update (DU) bit is defined as follows:

- a) DU set to zero indicates that the device server shall update the log parameter value to reflect all events that should be noted by that parameter; or
- b) DU set to one indicates that the device server shall not update the log parameter value except in response to a LOG SELECT command that specifies a new value for the parameter.

NOTE 44 - When updating cumulative log parameter values, a device server may use volatile memory to hold these values until a LOG SELECT or LOG SENSE command is received with an SP bit set to one or a vendor specific event occurs. As a result the updated cumulative log parameter values may be lost if a power cycle occurs.

The device server shall ignore the DU bit for threshold values, indicated by the PC field (see table 140 in 6.5.1) of the LOG SENSE command, or for list parameters as indicated by the FORMAT AND LINKING field received with a LOG SELECT command. The device server shall ignore the value of the DU bit in any such log parameters received with a LOG SELECT command.

A target save disable (TSD) bit set to zero indicates that the logical unit implicitly saves the log parameter at vendor specific intervals. This implicit saving operation shall be done frequently enough to insure that the cumulative parameter values retain statistical significance (i.e., across power cycles). A TSD bit set to one indicates that either the logical unit does not implicitly save the log parameter or implicit saving of the log parameter has been disabled individually by an application client setting the TSD bit to one. An application client may disable the implicit saving for all log parameters without changing any TSD bits using the GLTSD bit in the Control mode page (see 7.4.6).

An enable threshold comparison (ETC) bit set to one indicates that a comparison to the threshold value is performed whenever the cumulative value is updated. An ETC bit set to zero indicates that a comparison is not performed. The value of the ETC bit is the same for cumulative and threshold parameters.

The threshold met criteria (TMC) field (see table 285) defines the basis for comparison of the cumulative and threshold values. The TMC field is valid only if the ETC bit is set to one. The value of the TMC field is the same for cumulative and threshold parameters.

**Table 285 — Threshold met criteria (TMC) field**

Code	Basis for comparison
00b	Every update of the cumulative value
01b	Cumulative value equal to threshold value
10b	Cumulative value not equal to threshold value
11b	Cumulative value greater than threshold value

If the ETC bit is set to one and the result of the comparison is true, the device server shall establish a unit attention condition (see SAM-4) for the initiator port associated with every I\_T nexus, with the additional sense code set to THRESHOLD CONDITION MET.

The FORMAT AND LINKING field (see table 286) indicates the type of log parameter and how parameters that reach their maximum value are handled.

**Table 286 — FORMAT AND LINKING field**

Code	Log parameter type	Maximum value handling
00b	Data counter	If any other parameter in this log page reaches its maximum value, then this parameter shall stop incrementing until reinitialized by a LOG SELECT command.
01b	List format ASCII data (see 4.4.1)	No maximum values to handle
10b	Data counter	If another parameter reported in this log page reaches its maximum value, then this parameter shall not stop incrementing. This parameter may be reinitialized by a LOG SELECT command.
11b	List format binary data	No maximum values to handle

A FORMAT AND LINKING field set to 00b or 10b indicates that the parameter is a data counter. Data counters are saturating counters associated with one or more events. A data counter is incremented whenever one of these events occurs. If a data counter has associated with it a vendor specific maximum value, then upon reaching this maximum value, the data counter shall not be incremented (i.e., its value does not wrap). When a data counter reaches its maximum value, the device server shall set the associated DU bit to one and handle other data counters in the log page as defined in table 286. If the data counter is at or reaches its maximum value during the processing of a command, the device server shall complete the command. If the command completes without error, except for the data counter being at its maximum value, and if the RLEC bit of the Control mode page (see 7.4.6) is set to one, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to RECOVERED ERROR, and the additional sense code set to LOG COUNTER AT MAXIMUM.

A FORMAT AND LINKING field set to 01b or 11b indicates that the parameter is a list parameter. If the FORMAT AND LINKING field set to 01b or 11b, the ETC bit and TMC field shall be set to zero. If the FORMAT AND LINKING field set to 01b or 11b and either the ETC bit and TMC field shall is set to a non-zero value in LOG SELECT parameter data, then the device server shall terminate the command with CHECK CONDITION status and shall set the sense key to ILLEGAL REQUEST and the additional sense code to INVALID FIELD IN PARAMETER LIST.

If more than one list parameter is defined in a single log page, the following rules apply to assigning parameter codes:

- a) The parameter updated last shall have a higher parameter code than the parameter updated previously, except as defined in rule b); and
- b) When the maximum parameter code value supported by the logical unit is reached, the device server shall assign the lowest parameter code value to the next log parameter (i.e., wrap-around parameter codes). If the associated LOG SELECT command completes without error, except for the parameter code being at its maximum value, and if the RLEC bit of the Control mode page (see 7.4.6) is set to one, then the command shall be terminated with CHECK CONDITION status, with the sense key set to RECOVERED ERROR, and the additional sense code set to LOG LIST CODES EXHAUSTED.

NOTE 45 - List parameters may be used to store the locations of defective blocks in the following manner. When a defective block is identified, a list parameter is updated to reflect the location and cause of the defect. When the next defect is encountered, the list parameter with the next higher parameter code is updated to record this defect. The size of the log page may be made vendor specific to accommodate memory limitations. It is recommended that one or more data counter parameters be defined for the log page to keep track of the number of valid list parameters and the parameter code of the parameter with the oldest recorded defect. This technique may be adapted to record other types of information.

If a LOG SELECT command's parameter data contains a FORMAT AND LINKING value that is not allowed (see table 287) based on the FORMAT AND LINKING field value returned by a LOG SENSE command, the LOG SELECT command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

**Table 287 — Allowed LOG SELECT FORMAT AND LINKING field values**

Format and linking values returned by LOG SENSE	LOG SELECT format and linking values			
	00b	01b	10b	11b
00b	Allowed	Not Allowed	Allowed	Not Allowed
01b	Not Allowed	Allowed	Not Allowed	Not Allowed
10b	Allowed	Not Allowed	Allowed	Not Allowed
11b	Not Allowed	Not Allowed	Not Allowed	Allowed

The value for the FORMAT AND LINKING field for each log parameter are defined in the subclause that describes each log parameter.

If the value in the FORMAT AND LINKING field is 00b or 10b (i.e., the log parameter is a data counter parameter), then the values for the bits and fields in the parameter control byte for a LOG SENSE command or LOG SELECT command are described in 7.2.1.2.2.2.

If the value in the FORMAT AND LINKING field is 01b or 11b (i.e., the log parameter is a list parameter), then the values for the bits and fields in the parameter control byte for a LOG SENSE command or LOG SELECT command are described in 7.2.1.2.2.3.

### 7.2.1.2.2.2 Parameter control byte values for data counter parameters

If the FORMAT AND LINKING field is set to 00b or 10b (i.e., the parameter is a data counter parameter), the values for the bits and fields in the parameter control byte are shown in table 288.

**Table 288 — Parameter control byte values for data counter parameters**

Field or bit	Value for LOG SENSE	Value for LOG SELECT	Description
DU	0 or 1	0 or 1	When the DU bit is set to zero, the device server shall update the log parameter value to reflect all events that should be noted by that parameter. When the DU bit is set to one, the device server shall not update the log parameter value except in response to a LOG SELECT command that specifies a new value for the parameter.
TSD	0 or 1	0 or 1	When the TSD bit is set to zero, the device server shall save the log parameter to its medium at vendor specific intervals. When the TSD bit is set to one, implicit saving of the log parameter is disabled by an application client.
ETC	0 or 1	0 or 1	When the ETC bit is set to one, a comparison to the threshold value is performed whenever the cumulative value is updated. When the ETC bit is set to zero, a comparison is not performed.
TMC	any	any	The TMC field (see table 285 in 7.2.1.2.2.1) defines the basis for comparison of the cumulative and threshold values. The TMC field is valid only if the ETC bit is set to one.
FORMAT AND LINKING	00b or 10b	00b or 10b	The log parameter is a data counter (see table 286 in 7.2.1.2.2.1).

### 7.2.1.2.2.3 Parameter control byte values for list parameters

If the FORMAT AND LINKING field is set to 01b or 11b (i.e., the parameter is a list parameter), the values for the bits and fields in the parameter control byte are shown in table 289.

**Table 289 — Parameter control byte values for list parameters**

Field or bit	Value for LOG SENSE	Value for LOG SELECT	Description
DU	0 or 1	ignored	The DU bit is not defined for list parameters. The DU bit shall be set to zero when read with the LOG SENSE command, and shall be ignored when written with the LOG SELECT command.
TSD	0 or 1	0 or 1	When the TSD bit is set to zero, the device server shall save the log parameter to its medium at vendor specific intervals. When the TSD bit is set to one, implicit saving of the log parameter is disabled by an application client.
ETC	0 or 1	ignored	The ETC bit is not defined for list parameters. The ETC bit shall be set to zero when read with the LOG SENSE command, and shall be ignored when written with the LOG SELECT command.
TMC	any	ignored	The TMC field is not defined for list parameters. The TMC field shall be set to zero when read with the LOG SENSE command, and shall be ignored when written with the LOG SELECT command.
FORMAT AND LINKING	01b	01b	The log parameter is an ASCII format list parameter (see table 286 in 7.2.1.2.2.1).
	11b	11b	The log parameter is an binary format list parameter (see table 286 in 7.2.1.2.2.1).

### 7.2.2 Log page codes for all device types

The page code assignments for the log pages are listed in table 290.

**Table 290 — Log page codes**

Page Code <sup>a</sup>	Subpage Code <sup>a</sup>	Log Page Name	Reference
0Fh	00h	Application Client	7.2.3
01h	00h	Buffer Over-Run/Under-Run	7.2.4
19h	00h	General Statistics and Performance	7.2.13
19h	01h to 1Fh	Group Statistics and Performance (1 to 31)	7.2.13
2Fh	00h	Informational Exceptions	7.2.6
0Bh	00h	Last <i>n</i> Deferred Errors or Asynchronous Events	7.2.7
07h	00h	Last <i>n</i> Error Events	7.2.8
06h	00h	Non-Medium Error	7.2.9
18h	00h to FEh	Protocol Specific Port <sup>b</sup>	7.2.10
03h	00h	Read Error Counter	7.2.5
04h	00h	Read Reverse Error Counter	7.2.5
10h	00h	Self-Test Results	7.2.11
0Eh	00h	Start-Stop Cycle Counter	7.2.12
00h	00h	Supported Log Pages	7.2.14
00h	FFh	Supported Log Pages and Subpages	7.2.15
01h to 3Fh	FFh	Supported Subpages	7.2.16
0Dh	00h	Temperature	7.2.17
05h	00h	Verify Error Counter	7.2.5
02h	00h	Write Error Counter	7.2.5
08h to 0Ah	00h to FEh	Reserved (may be used by specific device types)	
0Ch	00h to FEh	Reserved (may be used by specific device types)	
11h to 17h	00h to FEh	Reserved (may be used by specific device types)	
19h	20h to FEh	Reserved	
1Ah to 2Eh	00h to FEh	Reserved (may be used by specific device types)	
3Fh	00h to FEh	Reserved	
30h to 3Eh	00h to FEh	Vendor specific	
Annex D contains a listing of log pages codes and subpage codes in numeric order.			
<sup>a</sup> All page code and subpage code combinations not shown in this table are reserved.			
<sup>b</sup> Each SCSI transport protocol standard (see 3.1.140) may define a different name for these log pages.			

### 7.2.3 Application Client log page

The Application Client log page (see table 291) provides a place for application clients to store information. The page code for the application client page is 0Fh.

**Table 291 — Application client log page**

Bit Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (0Fh)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						
3							(LSB)	
	Application client log parameters							
4	Application client log parameter [first]							
	⋮							
	Application client log parameter [last]							
n								

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.2.1.

Parameter codes 0000h to 0FFFh are for general usage application client data. The intended use for this information is to aid in describing the system configuration and system problems, but the specific definition of the data is application client specific. The general usage application client data parameters all have the format shown in table 292.

**Table 292 — General usage application client parameter data**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1	PARAMETER CODE (LSB)							
2	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (FCh)							
4	GENERAL USAGE PARAMETER BYTES							
255								

For general usage application client data, the value in the PARAMETER CODE field shall be between 0000h and 0FFFh. The first supported general usage application client parameter code shall be 0000h and additional supported parameters shall be sequentially numbered. If any general usage parameter codes are implemented, the device shall support at least 64 general usage parameter descriptors and they shall be parameter codes 0000h to 003Fh.

For the general usage application client parameter, the PARAMETER LENGTH value for each parameter shall be FCh.

The FORMAT AND LINKING field for log parameters 0000h through 0FFFh in the Application Client log page shall be set to 11b, indicating that the parameters are binary format list parameters. The values of the bits and fields in the parameter control byte for binary format list parameters are described in 7.2.1.2.2.3.

The values stored in the GENERAL USAGE PARAMETER BYTES represent data sent to the device server in a previous LOG SELECT command. If a previous LOG SELECT command has not occurred, the data is vendor specific.

In the application client log page, parameter codes 1000h through FFFFh are reserved.

#### 7.2.4 Buffer Over-Run/Under-Run log page

The Buffer Over-Run/Under-Run log page (page code 01h) defines 24 data counters that may be used to record the number of buffer over-runs or under-runs for the logical unit. A logical unit that implements this log page may implement one or more of the defined data counters.

A buffer over-run or under-run may occur when a SCSI initiator device does not transmit data to or from the logical unit's buffer fast enough to keep up with reading or writing the media. A buffer over-run condition may occur during a read operation when a buffer full condition prevents continued transfer of data from the media to the buffer. A buffer under-run condition may occur during a write operation when a buffer empty condition prevents continued transfer of data to the media from the buffer. Most devices incur a delay at this point while the media is repositioned.

The FORMAT AND LINKING field for each log parameter in the Buffer Over-Run/Under-Run log page shall be set to 00b or 10b indicating that the log parameters are data counter log parameters. The values of the bits and fields in the parameter control byte for data counter parameters are described in 7.2.1.2.2.2.

Table 293 defines the PARAMETER CODE field for the buffer over-run/under-run counters.

**Table 293 — Parameter code field for buffer over-run/under-run counters**

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved							
1	COUNT BASIS				CAUSE			TYPE

The PARAMETER CODE field for buffer over-run/under-run counters contains a 16-bit value comprised of eight reserved bits, a COUNT BASIS field (see table 294), a CAUSE field (see table 295), and a TYPE bit. These are concatenated to determine the value of the parameter code for that log parameter. (E.g., a counter for parameter code value of 0023h specifies a count basis of 001b, a cause of 0001b, and a type of 1b. This counter is incremented once per command that experiences an over-run due to a service delivery subsystem being busy.)

The COUNT BASIS field defines the criteria for incrementing the counter. The criteria are defined in table 294.

**Table 294 — COUNT BASIS field**

Code	Description
000b	Undefined
001b	Per command
010b	Per I_T nexus loss
011b	Per unit of time
100b to 111b	Reserved



NOTE 46 - The per unit of time count basis is device type specific. Direct access block devices typically use a latency period (i.e., one revolution of the medium) as the unit of time.

The CAUSE field indicates the reason that the buffer over-run or under-run occurred. The following causes are defined in table 295.

**Table 295 — CAUSE field**

Code	Description
0h	Undefined
1h	Service delivery subsystem busy
2h	Transfer rate too slow
3h to Fh	Reserved

The TYPE bit indicates whether the counter records buffer under-runs or over-runs. A TYPE bit set to zero specifies a buffer under-run condition and a TYPE bit set to one specifies a buffer over-run condition.

Each counter contains the total number of times buffer over-run or under-run conditions have occurred since the last time the counter was cleared. The counter shall be incremented for each occurrence of a buffer under-run or over-run condition and may be incremented more than once for multiple occurrences during the processing of a single command.

## 7.2.5 Error Counter log pages

This subclause defines the Error Counter log pages (see table 296).

**Table 296 — Error Counter log page codes**

Page Code	Log Page Name
03h	Read Error Counter
04h	Read Reverse Error Counter
05h	Verify Error Counter
02h	Write Error Counter

The log page format is defined in 7.2.1. A log page may return one or more log parameters that report data counters for events defined by the parameter codes. Table 297 defines the parameter codes for the Error Counter log pages.

**Table 297 — Parameter codes for Error Counter log pages**

Parameter code	Description
0000h	Errors corrected without substantial delay
0001h	Errors corrected with possible delays
0002h	Total (e.g., rewrites or rereads)
0003h	Total errors corrected
0004h	Total times correction algorithm processed
0005h	Total bytes processed
0006h	Total uncorrected errors
0007h to 7FFFh	Reserved
8000h to FFFFh	Vendor specific

NOTE 47 - The exact definition of the error counters is not part of this standard. These counters should not be used to compare products because the products may define errors differently.

The FORMAT AND LINKING field for each log parameter in the Error Counter log pages shall be set to 00b or 10b indicating that the log parameters are data counter log parameters. The values of the bits and fields in the parameter control byte for data counter parameters are described in 7.2.1.2.2.2.

### 7.2.6 Informational Exceptions log page

The Informational Exceptions log page (see table 298) provides a place for reporting detail about informational exceptions. The page code for the Informational Exceptions log page is 2Fh.

**Table 298 — Informational Exceptions log page**

Bit Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (2Fh)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						(LSB)
3								
	Informational exceptions log parameters							
4	Informational exceptions log parameter [first]							
	⋮							
n	Informational exceptions log parameter [last]							

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.2.1.

Table 299 defines the parameter codes.

**Table 299 — Informational exceptions parameter codes**

Parameter code	Description
0000h	Informational exceptions general parameter data
0001h to FFFFh	Vendor specific

The informational exceptions general parameter data page has the format shown in table 300.

**Table 300 — Informational exceptions general parameter data**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____							
1	PARAMETER CODE (0000h) _____ (LSB)							
2	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (n-3)							

**Table 300 — Informational exceptions general parameter data**

Bit Byte	7	6	5	4	3	2	1	0
4	INFORMATIONAL EXCEPTION ADDITIONAL SENSE CODE							
5	INFORMATIONAL EXCEPTION ADDITIONAL SENSE CODE QUALIFIER							
6	MOST RECENT TEMPERATURE READING							
7	Vendor specific							
n								

The FORMAT AND LINKING field for log parameter 0000h in the Informational Exception log page shall be set to 11b, indicating that the parameter is a binary format list parameter. The values of the bits and fields in the parameter control byte for binary format list parameters are described in 7.2.1.2.2.3.

The PARAMETER LENGTH field is described in 7.2.1. The parameter length shall be at least 04h.

If the INFORMATIONAL EXCEPTION ADDITIONAL SENSE CODE field contains zero, no informational exception condition is pending and contents of the INFORMATIONAL EXCEPTION ADDITIONAL SENSE CODE QUALIFIER field are unspecified. If the INFORMATIONAL EXCEPTION ADDITIONAL SENSE CODE field contains any value other than zero, an informational exception condition exists that has an additional sense code indicated by INFORMATIONAL EXCEPTION ADDITIONAL SENSE CODE field and an ADDITIONAL SENSE CODE QUALIFIER indicated by the INFORMATIONAL EXCEPTION ADDITIONAL SENSE CODE QUALIFIER field.

The MOST RECENT TEMPERATURE READING field indicates the temperature in degrees Celsius of the SCSI target device at the time the LOG SENSE command is performed. Temperatures equal to or less than zero degrees Celsius shall be indicated by a value of zero. If the device server is unable to detect a valid temperature because of a sensor failure or other condition, the value returned shall be FFh. The temperature should be reported with an accuracy of plus or minus three Celsius degrees while the device is operating at a steady state within the environmental limits specified for the device.

### 7.2.7 Last *n* Deferred Errors or Asynchronous Events log page

The Last *n* Deferred Errors or Asynchronous Events log page (page code 0Bh) provides for a number of deferred errors or asynchronous events sense data records using the list parameter format of the log page. The number of these deferred errors or asynchronous events records supported, *n*, is vendor specific. Each deferred error or asynchronous event record contains SCSI sense data for a deferred error or asynchronous event that has occurred. The parameter code associated with the record indicates the relative time at which the deferred error or asynchronous event occurred. A higher parameter code indicates that the deferred error or asynchronous event occurred later in time.

The content of the PARAMETER VALUE field of each log parameter is the SCSI sense data describing the deferred error.

The FORMAT AND LINKING field for each log parameter in the Last *n* Deferred Errors or Asynchronous Events log page shall be set to 11b, indicating that the parameters are binary format list parameters. The values of the bits and fields in the parameter control byte for binary format list parameters are described in 7.2.1.2.2.3.

### 7.2.8 Last *n* Error Events log page

The Last *n* Error Events log page (page code 07h) provides for a number of error-event records using the list parameter format of the log page. The number of these error-event records supported, *n*, is vendor specific. Each error-event record contains vendor specific diagnostic information for a single error encountered by the device. The

parameter code associated with error-event record indicates the relative time at which the error occurred. A higher parameter code indicates that the error event occurred later in time.

The content of the **PARAMETER VALUE** field of each log parameter is ASCII data (see 4.4.1) that may describe the error event. The contents of the character string is not defined by this standard.

When the last supported parameter code is used by an error-event record, the recording on this log page of all subsequent error information shall cease until one or more of the list parameters with the highest parameter codes have been reinitialized. If the **RLEC** bit of the Control mode page (see 7.4.6) is set to one, the command shall be terminated with **CHECK CONDITION** status, with the sense key set to **RECOVERED ERROR**, and the additional sense code set to **LOG LIST CODES EXHAUSTED**.

The **FORMAT AND LINKING** field for each log parameter in the Last *n* Error Events log page shall be set to 01b, indicating that the parameters are ASCII format list parameters. The values of the bits and fields in the parameter control byte for ASCII format list parameters are described in 7.2.1.2.2.3.

### 7.2.9 Non-Medium Error log page

The Non-Medium Error log page (page code 06h) provides for summing the occurrences of recoverable error events other than write, read, or verify failures. No discrimination among the various types of events is provided by parameter code (see table 301). Vendor specific discrimination may be provided through the vendor specific parameter codes.

**Table 301 — Non-medium error event parameter codes**

Parameter code	Description
0000h	Non-medium error count
0001h to 7FFFh	Reserved
8000h to FFFFh	Vendor specific error counts

The **FORMAT AND LINKING** field for each log parameter in the Non-Medium Error log page shall be set to 00b or 10b indicating that the log parameters are data counter log parameters. The values of the bits and fields in the parameter control byte for data counter parameters are described in 7.2.1.2.2.2.

### 7.2.10 Protocol Specific Port log page

The Protocol Specific Port log page (see table 302) provides SCSI transport protocol specific parameters that are associated with the SCSI target ports in the SCSI target device. This log page may be implemented in any logical unit, including the **TARGET LOG PAGES** well-known logical unit (see 8.4). See the SCSI transport protocol standard (see 3.1.113) for definitions of the protocol specific log parameters.

**Table 302 — Protocol Specific Port log page**

Bit Byte	7	6	5	4	3	2	1	0
0	DS	SPF	PAGE CODE (18h)					
1	SUBPAGE CODE (00h to FEh)							
2	(MSB)	PAGE LENGTH (n-3)						
3								
	Protocol specific port log parameters							
4	Protocol specific port log parameter [first]							
	⋮							
	Protocol specific port log parameter [last]							
n								

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.2.1.

Table 303 shows the format of a protocol specific port log parameter.

**Table 303 — Protocol specific port log parameter format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____ PARAMETER CODE (relative target port identifier) _____ (LSB)							
1								
2	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (x-3)							
4	Reserved				PROTOCOL IDENTIFIER			
5								
x	SCSI transport protocol specific _____							

The PARAMETER CODE field contains the relative target port identifier (see 3.1.120) of the target port for which the parameter data applies.

The FORMAT AND LINKING field for each log parameter in the Protocol Specific Port log page shall be set to 11b, indicating that the parameters are binary format list parameters. The values of the bits and fields in the parameter control byte for binary format list parameters are described in 7.2.1.2.2.3.

The PARAMETER LENGTH field indicates the number of bytes remaining in the log parameter.

The PROTOCOL IDENTIFIER field contains one of the values shown in table 361 (see 7.5.1) to identify the SCSI transport protocol standard that defines the SCSI transport protocol specific data in this log parameter. The SCSI transport protocol specific data is defined by the corresponding SCSI transport protocol standard.

### 7.2.11 Self-Test Results log page

The Self-Test Results log page (see table 304) provides the results from the 20 most recent self-tests (see 5.6). Results from the most recent self-test or the self-test currently in progress shall be reported in the first self-test log parameter; results from the second most recent self-test shall be reported in the second self-test log parameter; etc. If fewer than 20 self-tests have occurred, the unused self-test log parameter entries shall be zero filled.

**Table 304 — Self-Test Results log page**

Bit Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (10h)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (190h)						
3								
	(LSB)							
	Self-test results log parameters							
4	Self-test results log parameter [first]							
23								
	⋮							
384	Self-test results log parameter [twentieth]							
403								

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.2.1.

Table 305 shows the format of one self-test log parameter.

**Table 305 — Self-test results log parameter format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____							
1	PARAMETER CODE (0001h to 0014h) _____ (LSB)							
2	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (10h)							
4	SELF-TEST CODE			Reserved		SELF-TEST RESULTS		
5	SELF-TEST NUMBER							
6	(MSB) _____							
7	ACCUMULATED POWER ON HOURS _____ (LSB)							
8	(MSB) _____							
15	ADDRESS OF FIRST FAILURE _____ (LSB)							
16	Reserved				SENSE KEY			
17	ADDITIONAL SENSE CODE							
18	ADDITIONAL SENSE CODE QUALIFIER							
19	Vendor specific							

The PARAMETER CODE field identifies the log parameter being transferred. The PARAMETER CODE field for the results of the most recent self-test shall contain 0001h; the PARAMETER CODE field for the results of the second most recent test shall contain 0002h; etc.

The FORMAT AND LINKING field for each log parameter in the Self-Test Results log page shall be set to 11b, indicating that the parameters are binary format list parameters. The values of the bits and fields in the parameter control byte for binary format list parameters are described in 7.2.1.2.2.3.

The PARAMETER LENGTH field shall contain 10h.

The SELF-TEST CODE field contains the value in the SELF-TEST CODE field of the SEND DIAGNOSTIC command that initiated this self-test (see 6.32).

Table 306 defines the content of the SELF-TEST RESULTS field.

**Table 306 — SELF-TEST RESULTS field**

Code	Description
0h	The self-test completed without error.
1h	The background self-test was aborted by the application client using a SEND DIAGNOSTIC command (see 6.32) with the SELF-TEST CODE field set to 100b (i.e., abort background self-test).
2h	The self-test routine was aborted by an application client using a method other than a SEND DIAGNOSTIC command with the SELF-TEST CODE field set to 100b (e.g., by a task management function, or by issuing an exception command as defined in 5.6.3).
3h	An unknown error occurred while the device server was processing the self-test and the device server was unable to complete the self-test.
4h	The self-test completed with a failure in a test segment, and the test segment that failed is not known.
5h	The first segment of the self-test failed.
6h	The second segment of the self-test failed.
7h	Another segment of the self-test failed and which test is indicated by the contents of the SELF-TEST NUMBER field.
8h to Eh	Reserved
Fh	The self-test is in progress.

The SELF-TEST NUMBER field identifies the self-test that failed and consists of either:

- a) The number of the segment that failed during the self-test; or
- b) The number of the test that failed and the number of the segment in which the test was run, using a vendor specific method for placing the two values in the one field.

When the segment in which the failure occurred is not able to be identified or need not be identified, the SELF-TEST NUMBER field shall contain 00h.

The ACCUMULATED POWER ON HOURS field contains the total hours the device server has been powered on since manufacturing at the time the self-test is completed. If the self-test is still in progress, the ACCUMULATED POWER ON HOURS field shall be set to zero. If the number of hours is greater than FFFFh, the ACCUMULATED POWER ON HOURS field shall be set to FFFFh.

The ADDRESS OF FIRST FAILURE field contains information that locates the failure on the media. If the logical unit implements logical blocks, the content of the ADDRESS OF FIRST FAILURE field is the first logical block address where a self-test error occurred. This implies nothing about the quality of any other logical block on the logical unit, since the testing during which the error occurred may not have been performed in a sequential manner. This value shall not change (e.g., as the result of block reassignment). The content of the ADDRESS OF FIRST FAILURE field shall be FFFF FFFF FFFF FFFFh if no errors occurred during the self-test or if the error that occurred is not related to an identifiable media address.

The SENSE KEY field, ADDITIONAL SENSE CODE field, and ADDITIONAL SENSE CODE QUALIFIER field may contain a hierarchy of additional information relating to error or exception conditions that occurred during the self-test represented in the same format used by the sense data (see 4.5).



### 7.2.12 Start-Stop Cycle Counter log page

This subclause defines the Start-Stop Cycle Counter log page (page code 0Eh). A device server that implements the Start-Stop Cycle Counter log page shall implement one or more of the defined parameters. Table 307 shows the Start-Stop Cycle Counter log page with all parameters present.

**Table 307 — Start-Stop Cycle Counter log page (part 1 of 2)**

Bit Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (0Eh)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (24h)						(LSB)
3								
4	(MSB)	PARAMETER CODE (0001h) Date of Manufacture						(LSB)
5								
6	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
7	PARAMETER LENGTH (06h)							
8	(MSB)	YEAR OF MANUFACTURE (4 ASCII characters)						(LSB)
11								
12	(MSB)	WEEK OF MANUFACTURE (2 ASCII characters)						(LSB)
13								
14	(MSB)	PARAMETER CODE (0002h) Accounting Date						(LSB)
15								
16	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
17	PARAMETER LENGTH (06h)							
18	(MSB)	ACCOUNTING DATE YEAR (4 ASCII characters)						(LSB)
21								
22	(MSB)	ACCOUNTING DATE WEEK (2 ASCII characters)						(LSB)
23								
24	(MSB)	PARAMETER CODE (0003h) Specified cycle count over device lifetime						(LSB)
25								
26	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
27	PARAMETER LENGTH (04h)							
28	(MSB)	SPECIFIED CYCLE COUNT OVER DEVICE LIFETIME (4-byte binary number)						(LSB)
31								

**Table 307 — Start-Stop Cycle Counter log page (part 2 of 2)**

Bit Byte	7	6	5	4	3	2	1	0
32	(MSB) _____							
33	PARAMETER CODE (0004h) Accumulated start-stop cycles _____ (LSB)							
34	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
35	PARAMETER LENGTH (04h)							
36	(MSB) _____							
39	ACCUMULATED START-STOP CYCLES (4-byte binary number) _____ (LSB)							

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.2.1.

The year and week in the year that the SCSI target device was manufactured shall be contained in the parameter value of the log parameter in which the parameter code is 0001h. The date is expressed in numeric ASCII characters (30h to 39h) in the form YYYYWW, as shown in table 307. If a LOG SELECT command attempts to change the value of the date of manufacture log parameter, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The FORMAT AND LINKING field for log parameter 0001h (i.e., the Date of Manufacturing parameter) in the Start-Stop Cycle Counter log page shall be set to 01b, indicating that the parameter is an ASCII format list parameter. The values of the bits and fields in the parameter control byte for ASCII format list parameters are described in 7.2.1.2.2.3.

The accounting date specified by parameter code 0002h may be saved using a LOG SELECT command to indicate when the device was placed in service. If the parameter is not yet set or is not settable, the default value placed in the parameter field shall be 6 ASCII space characters (20h). The field shall not be checked for validity by the device server.

The FORMAT AND LINKING field for log parameter 0002h (i.e., the Accounting Data parameter) in the Start-Stop Cycle Counter log page shall be set to 01b, indicating that the parameter is an ASCII format list parameter. The values of the bits and fields in the parameter control byte for ASCII format list parameters are described in 7.2.1.2.2.3.

The parameter value in the specified cycle count over device lifetime log parameter (parameter code 0003h) shall contain a four-byte binary value that indicates how many stop-start cycles may typically be performed over the lifetime of the SCSI target device without degrading the SCSI target device's operation or reliability outside the limits specified by the manufacturer of the SCSI target device. If a LOG SELECT command attempts to change the value of the specified cycle count over device lifetime log parameter, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The FORMAT AND LINKING field for log parameter 0003h (i.e., the Specified Cycle Count Over Device Lifetime parameter) in the Start-Stop Cycle Counter log page shall be set to 11b, indicating that the parameter is a binary format list parameter. The values of the bits and fields in the parameter control byte for binary format list parameters are described in 7.2.1.2.2.3.

The parameter value in the accumulated start-stop cycles log parameter (parameter code 0004h) shall contain a four-byte binary value that indicates how many stop-start cycles the SCSI target device has detected since its date of manufacture. The accumulated start-stop cycles counter is a saturating counter. If a LOG SELECT command attempts to change the value of the accumulated start-stop cycles log parameter, the command shall be terminated

with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST. The time at which the count is incremented during a start-stop cycle is vendor specific. For rotating magnetic storage devices, a single start-stop cycle is defined as an operational cycle that begins with the disk spindle at rest, continues while the disk accelerates to its normal operational rotational rate, continues during the entire period the disk is rotating, continues as the disk decelerates toward a resting state, and ends when the disk is no longer rotating. For devices without a spindle or with multiple spindles, the definition of a single start-stop cycle is vendor specific. The count is incremented by one for each complete start-stop cycle. No comparison with the value of parameter 0003h shall be performed by the device server.

The FORMAT AND LINKING field for log parameter 0004h (i.e., the Accumulated Start-Stop Cycles parameter) in the Start-Stop Cycle Counter log page shall be set to 11b, indicating that the parameter is a binary format list parameter. The values of the bits and fields in the parameter control byte for binary format list parameters are described in 7.2.1.2.2.3.

### 7.2.13 Statistics and Performance log pages

#### 7.2.13.1 Statistics and Performance log pages overview

The Statistics and Performance log pages consist of a General Statistics and Performance log page and up to 31 Group Statistics and Performance log pages. Each Group Statistics and Performance log pages only collects statistics and performance information for the group number specified in a read CDB or a write CDB.

The General Statistics and Performance log page (see 7.2.13.2) provides the following statistics and performance results associated to the addressed logical unit:

- a) Number of read commands;
- b) Number of write commands;
- c) Number of read logical blocks transmitted by a target port;
- d) Number of write logical blocks received by a target port;
- e) Read command processing time;
- f) Write command processing time;
- g) Sum of the command weights of the read commands plus write commands;
- h) Sum of the weighted command time of the read commands plus write commands;
- i) Idle time; and
- j) Time interval.

The Group Statistics and Performance log pages (see 7.2.13.3) provide the following statistics and performance results associated to the addressed logical unit and the GROUP NUMBER field:

- a) Number of read commands;
- b) Number of write commands;
- c) Number of read logical blocks transmitted by a target port;
- d) Number of write logical blocks received by a target port;
- e) Read command processing time; and
- f) Write command processing time.

In the Statistics and Performance log pages, read and write commands are those shown in table 308.

**Table 308 — Statistics and Performance log pages read and write commands**

Read commands <sup>a</sup>	Write commands <sup>a</sup>
READ(6)	WRITE(6)
READ(10)	WRITE(10)
READ(12)	WRITE(12)
READ(16)	WRITE(16)
READ(32)	WRITE(32)
	WRITE AND VERIFY(10)
	WRITE AND VERIFY(12)
	WRITE AND VERIFY(16)
	WRITE AND VERIFY(32)
<sup>a</sup> See SBC-3.	

### 7.2.13.2 General Statistics and Performance log page

Table 309 shows the General Statistics and Performance log page format.

**Table 309 — General Statistics and Performance log page**

Bit Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (19h)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (5Ch or 9Ch)						
3								(LSB)
	General Statistics and Performance log parameters							
4	Statistics and Performance log parameter (required)							
71	(see table 310)							
72	Idle Time log parameter (required)							
83	(see table 311)							
84	Time Interval log parameter (required)							
95	(see table 312)							
96	Force Unit Access Statistics and Performance log							
159	parameter (optional)						(see table 310)	

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.2.1.

Statistics and Performance log parameters not defined by this standard are reserved.

Table 310 shows the Statistics and Performance log parameter format.

**Table 310 — Statistics and Performance log parameter**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____							
1	PARAMETER CODE (0001h) _____ (LSB)							
2	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (40h) _____							
4	(MSB) _____							
11	NUMBER OF READ COMMANDS _____ (LSB)							
12	(MSB) _____							
19	NUMBER OF WRITE COMMANDS _____ (LSB)							
20	(MSB) _____							
27	NUMBER OF LOGICAL BLOCKS RECEIVED _____ (LSB)							
28	(MSB) _____							
35	NUMBER OF LOGICAL BLOCKS TRANSMITTED _____ (LSB)							
36	(MSB) _____							
43	READ COMMAND PROCESSING INTERVALS _____ (LSB)							
44	(MSB) _____							
51	WRITE COMMAND PROCESSING INTERVALS _____ (LSB)							
52	(MSB) _____							
59	WEIGHTED NUMBER OF READ COMMANDS PLUS WRITE COMMANDS _____ (LSB)							
60	(MSB) _____							
67	WEIGHTED READ COMMAND PROCESSING PLUS WRITE COMMAND PROCESSING _____ (LSB)							

The PARAMETER CODE field set to 0001h identifies the log parameter being transferred as the Statistics and Performance log parameter.

The FORMAT AND LINKING field for the Statistics and Performance log parameter in the Statistics and Performance log page shall be set to 10b, indicating that the parameter is a data counter parameter. The values of the bits and fields in the parameter control byte for a data counter parameter are defined in 7.2.1.2.2.2.

The PARAMETER LENGTH field specifies the length in bytes of the statistics and performance fields that follow.

The NUMBER OF READ COMMANDS field contains the number of read commands (see 7.2.13.1) received by the logical unit.

The NUMBER OF WRITE COMMANDS field contains the number of write commands (see 7.2.13.1) received by the logical unit.

The NUMBER OF LOGICAL BLOCKS RECEIVED field contains the number of logical blocks received by any SCSI target port for the logical unit as a result of write commands (see 7.2.13.1).

The NUMBER OF LOGICAL BLOCKS TRANSMITTED field contains the number of logical blocks transmitted by any SCSI target port for the logical unit as a result of read commands (see 7.2.13.1).

The READ COMMAND PROCESSING INTERVALS field contains the cumulative number of time intervals (see table 312) spent by the logical unit processing read commands (see 7.2.13.1).

The WRITE COMMAND PROCESSING INTERVALS field contains the cumulative number of time intervals (see table 312) spent by the logical unit processing write commands (see 7.2.13.1).

If command priority is supported (see SAM-4), then the WEIGHTED NUMBER OF READ COMMANDS PLUS WRITE COMMANDS field contains the cumulative command weight of the read commands and write commands (see 7.2.13.1) processed by the logical unit.

Command weight is calculated as follows:

$$\text{command weight} = (360\ 360 \div \text{command priority})$$

where:

command priority is as defined in SAM-4. However, if the computed command priority is zero, then the command priority shall be set to seven (i.e., a mid-range command priority value).

If command priority is not supported, then the WEIGHTED NUMBER OF READ COMMANDS PLUS WRITE COMMANDS field shall be set to zero.

If command priority is supported (see SAM-4), then the WEIGHTED READ COMMAND PROCESSING PLUS WRITE COMMAND PROCESSING field contains the cumulative weighted command time of the time intervals (see table 312) spent processing read commands and write commands (see 7.2.13.1) by the logical unit.

Weighted command time is calculated as follows:

$$\begin{aligned} \text{weighted command time} = & (\text{time increments processing the command} \times \text{time interval}) \\ & \times (360\ 360 \div \text{command priority}) \end{aligned}$$

where:

time increments processing a command is the number of time intervals from the time the task manager places the command into a task set until the device server sends a SCSI transport protocol service response for the command;

time interval is the value represented in the TIME INTERVAL DESCRIPTOR field of the Time Interval log parameter (see table 312), and

command priority is as defined in SAM-4. However, if the computed command priority is zero, then the command priority time shall be set to seven (i.e., a mid-range command priority value).

If command priority is not supported, then the WEIGHTED READ COMMAND PROCESSING PLUS WRITE COMMAND PROCESSING field shall be set to zero.

Table 311 shows the Idle Time log parameter format.

**Table 311 — Idle Time log parameter**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____							
1	PARAMETER CODE (0002h) _____ (LSB)							
2	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (08h) _____							
4	(MSB) _____							
11	IDLE TIME INTERVALS _____ (LSB)							

The PARAMETER CODE field set to 0002h identifies the log parameter being transferred as the Idle Time log parameter.

The FORMAT AND LINKING field for the Idle Time log parameter in the Statistics and Performance log page shall be set to 10b, indicating that the parameter is a data counter parameter. The values of the bits and fields in the parameter control byte for a data counter parameter are defined in 7.2.1.2.2.2.

The PARAMETER LENGTH field specifies the length in bytes of the IDLE TIME INTERVALS field that follows.

The IDLE TIME INTERVALS field contains the cumulative number of idle times spent when there are no tasks in the task set and there are no tasks being processed by the logical unit.

Idle time is calculated as follows:

$$\text{idle time} = (\text{time increments not processing commands} \times \text{time interval})$$

where:

time increments not processing commands is the number of time intervals when there are no commands in the task set and the device server has sent a SCSI transport protocol service response for all commands being processed (i.e., there are no commands to be processed or being processed); and

time interval is the value represented in the time interval descriptor of the Time Interval log parameter (see table 312).

Table 312 shows the Time Interval log parameter format.

**Table 312 — Time Interval log parameter**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1	PARAMETER CODE (0003h) (LSB)							
2	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (08h)							
4								
11	Time interval descriptor							

The PARAMETER CODE field set to 0003h identifies the log parameter being transferred as the Time Interval log parameter.

The FORMAT AND LINKING field for the Time Interval log parameter in the Statistics and Performance log page shall be set to 10b, indicating that the parameter is a data counter parameter. The values of the bits and fields in the parameter control byte for a data counter parameter are defined in 7.2.1.2.2.2.

The PARAMETER LENGTH field specifies the length in bytes of the IDLE TIME INTERVALS field that follows.

The time interval descriptor (see table 313) contains the time interval in seconds used in various time interval fields in the General Statistics and Performance log parameter (see table 310) and the Force Unit Access Statistics and Performance log parameter (see table 314).

**Table 313 — Time interval descriptor**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____							
3	EXPONENT _____ (LSB)							
4	(MSB) _____							
7	INTEGER _____ (LSB)							

The EXPONENT field contains the negative power of 10 exponent to multiply with the INTEGER field (e.g., a value of 9 represents  $10^{-9}$ )

When multiplied by the exponent, the INTEGER field contains the value that represents one time interval (e.g., a value of 5 in the INTEGER field and a value of 9 in the EXPONENT field represents a time interval of  $5 \times 10^{-9}$  seconds or 5 nanoseconds).



Table 314 shows the Force Unit Access Statistics and Performance log parameter format.

**Table 314 — Force Unit Access Statistics and Performance log parameter**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____							
1	PARAMETER CODE (0004h) _____ (LSB)							
2	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (40h) _____							
4	(MSB) _____							
11	NUMBER OF READ FUA COMMANDS _____ (LSB)							
12	(MSB) _____							
19	NUMBER OF WRITE FUA COMMANDS _____ (LSB)							
20	(MSB) _____							
27	NUMBER OF READ FUA_NV COMMANDS _____ (LSB)							
28	(MSB) _____							
35	NUMBER OF WRITE FUA_NV COMMANDS _____ (LSB)							
36	(MSB) _____							
43	READ FUA COMMAND PROCESSING INTERVALS _____ (LSB)							
44	(MSB) _____							
51	WRITE FUA COMMAND PROCESSING INTERVALS _____ (LSB)							
52	(MSB) _____							
59	READ FUA_NV COMMAND PROCESSING INTERVALS _____ (LSB)							
60	(MSB) _____							
67	WRITE FUA_NV COMMAND PROCESSING INTERVALS _____ (LSB)							

The PARAMETER CODE field set to 0004h identifies the log parameter being transferred as the Force Unit Access Statistics and Performance log parameter.

The FORMAT AND LINKING field for the Force Unit Access Statistics and Performance log parameter in the Statistics and Performance log page shall be set to 10b, indicating that the parameter is a data counter parameter. The values of the bits and fields in the parameter control byte for a data counter parameter are defined in 7.2.1.2.2.2.

The PARAMETER LENGTH field specifies the length in bytes of the statistics and performance fields that follow.

The NUMBER OF READ FUA COMMANDS field contains the number of read commands (see 7.2.13.1) with the FUA bit set to one received by the logical unit.

The NUMBER OF WRITE FUA COMMANDS field contains the number of write commands (see 7.2.13.1) with the FUA bit set to one received by the logical unit.

The NUMBER OF READ FUA\_NV COMMANDS field contains the number of read commands (see 7.2.13.1) with the FUA\_NV bit set to one received by the logical unit.

The NUMBER OF WRITE FUA\_NV COMMANDS field contains the number of write commands (see 7.2.13.1) with the FUA\_NV bit set to one received by the logical unit.

The READ FUA COMMAND PROCESSING INTERVALS field contains the cumulative number of time intervals (see table 312) spent by the logical unit processing read commands (see 7.2.13.1) with the FUA bit set to one.

The WRITE FUA COMMAND PROCESSING INTERVALS field contains the cumulative number of time intervals (see table 312) spent by the logical unit processing write commands (see 7.2.13.1) with the FUA bit set to one.

The READ FUA\_NV COMMAND PROCESSING INTERVALS field contains the cumulative number of time intervals (see table 312) spent by the logical unit processing read commands (see 7.2.13.1) with the FUA\_NV bit set to one.

The WRITE FUA\_NV COMMAND PROCESSING INTERVALS field contains the cumulative number of time intervals (see table 312) spent by the logical unit processing write commands (see 7.2.13.1) with the FUA\_NV bit set to one.

### 7.2.13.3 Group Statistics and Performance (n) log page

The Group Statistics and Performance (n) log pages (see table 315) provide logging of statistics and performance of read and write operations based on group numbers. There are 31 Group Statistics and Performance (n) log pages one for each group number. The statistics and performance information associated with each group number is collected in the corresponding Group Statistics and Performance (n) log page (e.g., operations associated with group number 16 are logged in the Group Statistics and Performance (16) log page).

**Table 315 — Group Statistics and Performance (n) log page**

Bit Byte	7	6	5	4	3	2	1	0
0	DS	SPF (1b)	PAGE CODE (19h)					
1	SUBPAGE CODE (01h to 1Fh) <sup>a</sup>							
2	(MSB)	PAGE LENGTH (34h or 74h)						
3								
	(LSB)							
Group Statistics and Performance log parameters								
4	Group n Statistics and Performance log parameter (required) (see table 317)							
55								
56	Group n Force Unit Access Statistics and Performance log parameter (optional) (see table 318)							
119								

<sup>a</sup> The log parameter associated with the specific group number as specified by the value of n is collected in the corresponding log parameter (e.g., the count of read commands with the GROUP NUMBER field set to 9 is logged in the GROUP N NUMBER OF READ COMMANDS field in the Group n Statistics and Performance log parameter of the Group Statistics and Performance (9) log page).

The DS bit, SPF bit, PAGE CODE field, and PAGE LENGTH field are described in 7.2.1.

Group n Statistics and Performance log parameters not defined by this standard are reserved.

The SUBPAGE CODE field is described in table 316.

**Table 316 — Group Statistics and Performance (n) subpage codes**

Subpage code	Log page name <sup>a</sup>	Group number <sup>b</sup>
01h	Group Statistics and Performance (1)	00001b
02h	Group Statistics and Performance (2)	00010b
03h	Group Statistics and Performance (3)	00011b
04h	Group Statistics and Performance (4)	00100b
05h	Group Statistics and Performance (5)	00101b
06h	Group Statistics and Performance (6)	00110b
07h	Group Statistics and Performance (7)	00111b
08h	Group Statistics and Performance (8)	01000b
09h	Group Statistics and Performance (9)	01001b
0Ah	Group Statistics and Performance (10)	01010b
0Bh	Group Statistics and Performance (11)	01011b
0Ch	Group Statistics and Performance (12)	01100b
0Dh	Group Statistics and Performance (13)	01101b
0Eh	Group Statistics and Performance (14)	01110b
0Fh	Group Statistics and Performance (15)	01111b
10h	Group Statistics and Performance (16)	10000b
11h	Group Statistics and Performance (17)	10001b
12h	Group Statistics and Performance (18)	10010b
13h	Group Statistics and Performance (19)	10011b
14h	Group Statistics and Performance (20)	10100b
15h	Group Statistics and Performance (21)	10101b
16h	Group Statistics and Performance (22)	10110b
17h	Group Statistics and Performance (23)	10111b
18h	Group Statistics and Performance (24)	11000b
19h	Group Statistics and Performance (25)	11001b
1Ah	Group Statistics and Performance (26)	11010b
1Bh	Group Statistics and Performance (27)	11011b
1Ch	Group Statistics and Performance (28)	11100b
1Dh	Group Statistics and Performance (29)	11101b
1Eh	Group Statistics and Performance (30)	11110b
1Fh	Group Statistics and Performance (31)	11111b

<sup>a</sup> The statistics and performance information associated with a group number is collected in the corresponding Group Statistics and Performance (n) log page (e.g., operations associated with group number 10000b are logged in the Group Statistics and Performance (16) log page).

<sup>b</sup> The GROUP NUMBER field is from the read command CDB or the write command CDB (see table 308 in 7.2.13.1 and SBC-3).

Table 317 shows the format of Group n Statistics and Performance log parameter.

**Table 317 — Group n Statistics and Performance log parameter**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____							
1	PARAMETER CODE (0001h) _____ (LSB)							
2	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (30h) _____							
4	(MSB) _____							
11	GROUP N NUMBER OF READ COMMANDS _____ (LSB)							
12	(MSB) _____							
19	GROUP N NUMBER OF WRITE COMMANDS _____ (LSB)							
20	(MSB) _____							
27	GROUP N NUMBER OF LOGICAL BLOCKS RECEIVED _____ (LSB)							
28	(MSB) _____							
35	GROUP N NUMBER OF LOGICAL BLOCKS TRANSMITTED _____ (LSB)							
36	(MSB) _____							
43	GROUP N READ COMMAND PROCESSING INTERVALS _____ (LSB)							
44	(MSB) _____							
51	GROUP N WRITE COMMAND PROCESSING INTERVALS _____ (LSB)							

The PARAMETER CODE field set to 0001h identifies the log parameter being transferred as the Group n Statistics and Performance log parameter.

The FORMAT AND LINKING field for the Group n Statistics and Performance log parameter in the Group Statistics and Performance (n) log pages shall be set to 10b, indicating that the parameter is a data counter parameter. The values of the bits and fields in the parameter control byte for a data counter parameter are defined in 7.2.1.2.2.2.

The PARAMETER LENGTH field specifies the length in bytes of the group n statistics and performance parameters that follow.

The GROUP N NUMBER OF READ COMMANDS field contains the number of read commands (see 7.2.13.1) received by the logical unit.

The GROUP N NUMBER OF WRITE COMMANDS field contains the number of write commands (see 7.2.13.1) received by the logical unit.

The GROUP N NUMBER OF LOGICAL BLOCKS RECEIVED field contains the number of logical blocks received by any SCSI target port for the logical unit as a result of write commands (see 7.2.13.1).

The GROUP N NUMBER OF LOGICAL BLOCKS TRANSMITTED field contains the number of logical blocks transmitted by any SCSI target port for the logical unit as a result of read commands (see 7.2.13.1).

The GROUP N READ COMMAND PROCESSING INTERVALS field contains the cumulative number of time intervals spent by the logical unit processing read commands (see 7.2.13.1). Time intervals are defined in the Time Interval log parameter (see table 312) in the General Statistics and Performance log page (see 7.2.13.2).

The GROUP N WRITE COMMAND PROCESSING INTERVALS field contains the cumulative number of time intervals spent by the logical unit processing write commands (see 7.2.13.1). Time intervals are defined in the Time Interval log parameter (see table 312) in the General Statistics and Performance log page (see 7.2.13.2).

Table 318 shows the format of Group n Force Unit Access Statistics and Performance log parameter.

**Table 318 — Group n Force Unit Access Statistics and Performance log parameter**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1	PARAMETER CODE (0004h) (LSB)							
2	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (40h)							
4	(MSB)							
11	GROUP N NUMBER OF READ FUA COMMANDS (LSB)							
12	(MSB)							
19	GROUP N NUMBER OF WRITE FUA COMMANDS (LSB)							
20	(MSB)							
27	GROUP N NUMBER OF READ FUA_NV COMMANDS (LSB)							
28	(MSB)							
35	GROUP N NUMBER OF WRITE FUA_NV COMMANDS (LSB)							
36	(MSB)							
43	GROUP N READ FUA COMMAND PROCESSING INTERVALS (LSB)							
44	(MSB)							
51	GROUP N WRITE FUA COMMAND PROCESSING INTERVALS (LSB)							
52	(MSB)							
59	GROUP N READ FUA_NV COMMAND PROCESSING INTERVALS (LSB)							
60	(MSB)							
67	GROUP N WRITE FUA_NV COMMAND PROCESSING INTERVALS (LSB)							

The PARAMETER CODE field set to 0004h identifies the log parameter being transferred as the Group n Force Unit Access Statistics and Performance log parameter.

The FORMAT AND LINKING field for the Group n Force Unit Access Statistics and Performance log parameter in the Group Statistics and Performance (n) log pages shall be set to 10b, indicating that the parameter is a data counter parameter. The values of the bits and fields in the parameter control byte for a data counter parameter are defined in 7.2.1.2.2.2.

The PARAMETER LENGTH field specifies the length in bytes of the group n statistics and performance parameters that follow.

The GROUP N NUMBER OF READ FUA COMMANDS field contains the number of read commands (see 7.2.13.1) with the FUA bit set to one received by the logical unit.

The GROUP N NUMBER OF WRITE FUA COMMANDS field contains the number of write commands (see 7.2.13.1) with the FUA bit set to one received by the logical unit.

The GROUP N NUMBER OF READ FUA\_NV COMMANDS field contains the number of read commands (see 7.2.13.1) with the FUA\_NV bit set to one received by the logical unit.

The GROUP N NUMBER OF WRITE FUA\_NV COMMANDS field contains the number of write commands (see 7.2.13.1) with the FUA\_NV bit set to one received by the logical unit.

The GROUP N READ FUA COMMAND PROCESSING INTERVALS field contains the cumulative number of time intervals spent by the logical unit processing read commands (see 7.2.13.1) with the FUA bit set to one. Time intervals are defined in the Time Interval log parameter (see table 312) in the General Statistics and Performance log page (see 7.2.13.2).

The GROUP N WRITE FUA COMMAND PROCESSING INTERVALS field contains the cumulative number of time intervals spent by the logical unit processing write commands (see 7.2.13.1) with the FUA bit set to one. Time intervals are defined in the Time Interval log parameter (see table 312) in the General Statistics and Performance log page (see 7.2.13.2).

The GROUP N READ FUA\_NV COMMAND PROCESSING INTERVALS field contains the cumulative number of time intervals spent by the logical unit processing read commands (see 7.2.13.1) with the FUA\_NV bit set to one. Time intervals are defined in the Time Interval log parameter (see table 312) in the General Statistics and Performance log page (see 7.2.13.2).

The GROUP N WRITE FUA\_NV COMMAND PROCESSING INTERVALS field contains the cumulative number of time intervals spent by the logical unit processing write commands (see 7.2.13.1) with the FUA\_NV bit set to one. Time intervals are defined in the Time Interval log parameter (see table 312) in the General Statistics and Performance log page (see 7.2.13.2).

#### 7.2.14 Supported Log Pages log page

The Supported Log Pages log page (see table 319) returns the list of log pages implemented by the logical unit. Logical units that implement the LOG SENSE command shall implement this log page.

**Table 319 — Supported Log Pages log page**

Bit Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (00h)					
1	SUBPAGE CODE (00h)							
2	(MSB) _____							
3	PAGE LENGTH (n-3)							(LSB)
	Supported pages							
4	Supported page descriptors							
n	_____							

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.2.1.

This log page is not defined for the LOG SELECT command. This log page returns the list of supported log pages for the specified logical unit.

The PAGE LENGTH field indicates the length in bytes of the following supported page descriptors.

The supported page descriptors shall contain a list of all log page codes (see table 320) with a subpage code of zero implemented by the logical unit in ascending order beginning with page code 00h.

**Table 320 — Supported page descriptor**

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved		PAGE CODE					

### 7.2.15 Supported Log Pages and Subpages log page

The Supported Log Pages and Subpages log page (see table 321) returns the list of log pages and subpages implemented by the logical unit. If log subpages are supported this page shall be supported.

**Table 321 — Supported Log Pages and Subpages log page**

Bit Byte	7	6	5	4	3	2	1	0
0	DS	SPF (1b)	PAGE CODE (00h)					
1	SUBPAGE CODE (FFh)							
2	(MSB) _____							
3	PAGE LENGTH (n-3) _____ (LSB)							
	Supported pages and subpages							
4	Supported page/subpage descriptors _____							
n								

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.2.1.

This log page is not defined for the LOG SELECT command. This log page returns the list of supported log pages and subpages for the specified logical unit.

The PAGE LENGTH field indicates the length in bytes of the following supported page and subpages descriptors.

The supported page/subpage descriptors (see table 322) shall be in ascending order sorted by page code then subpage code.

**Table 322 — Supported page/subpage descriptor**

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved		PAGE CODE					
1	SUBPAGE CODE							

The PAGE CODE field contains the number of the log page.

The SUBPAGE CODE field contains the subpage number of the log page.

### 7.2.16 Supported Subpages log page

The Supported Subpages log page (see table 323) returns the list of all subpage codes for a specified page code that are implemented by the logical unit. If log subpages are supported this page shall be supported.

**Table 323 — Supported Subpages log page**

Bit Byte	7	6	5	4	3	2	1	0
0	DS	SPF (1b)	PAGE CODE					
1	SUBPAGE CODE (FFh)							
2	(MSB)PAGE LENGTH (n-3)(LSB)							
3								
	Supported subpages							
4								
n	Supported page/subpage descriptors							

The DS bit, SPF bit, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.2.1.

The PAGE CODE field (see 7.2.1) indicates the log page for which subpage codes are being returned.

This log page is not defined for the LOG SELECT command. This log page returns the list of all subpage codes for a specified page code for the specified logical unit.

The PAGE LENGTH field indicates the length in bytes of the following supported page/subpages descriptors.

The supported page/subpage descriptors (see table 324) shall be in ascending order sorted by page code then subpage code.

**Table 324 — Supported page/subpage descriptor**

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved		PAGE CODE					
1	SUBPAGE CODE							

The PAGE CODE field contains the number of the log page.

NOTE 48 - The page code is the same in the page header (see table 323) and in each page/subpage descriptor (see table 324).

The SUBPAGE CODE field contains the subpage number of the log page.



### 7.2.17 Temperature log page

This subclause defines the Temperature log page (page code 0Dh). A device server that implements the Temperature log page shall implement parameter 0000h and may implement parameter 0001h. Table 325 shows the Temperature log page with all parameters present.

**Table 325 — Temperature log page**

Bit Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (0Dh)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (0Ch)						(LSB)
3								
4	(MSB)	PARAMETER CODE (0000h) Temperature						(LSB)
5								
6	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
7	PARAMETER LENGTH (02h)							
8	Reserved							
9	TEMPERATURE (degrees Celsius)							
10	(MSB)	PARAMETER CODE (0001h) Reference temperature						(LSB)
11								
12	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
13	PARAMETER LENGTH (02h)							
14	Reserved							
15	REFERENCE TEMPERATURE (degrees Celsius)							

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.2.1.

The parameter value in the Temperature log parameter (parameter code 0000h) shall contain a one-byte binary value that indicates the temperature of the SCSI target device in degrees Celsius at the time the LOG SENSE command is performed. Temperatures equal to or less than zero degrees Celsius shall be indicated by a value of zero. If the device server is unable to detect a valid temperature because of a sensor failure or other condition, then the value returned shall be FFh. The temperature should be reported with an accuracy of plus or minus three Celsius degrees while the SCSI target device is operating at a steady state within its environmental limits. No comparison is performed between the temperature value specified in parameter 0000h and the reference temperature specified in parameter 0001h.

The FORMAT AND LINKING field for log parameter 0000h (i.e., the Temperature parameter) in the Temperature log page shall be set to 11b, indicating that the parameter is a binary format list parameter. The values of the bits and fields in the parameter control byte for binary format list parameters are described in 7.2.1.2.2.3.

A reference temperature for the device may be returned by the device server as follows:

- a) If a reference temperature is returned, the parameter value in the reference temperature log parameter (parameter code 0001h) shall contain a one-byte binary value that indicates the maximum reported sensor temperature in degrees Celsius at which the SCSI target device is capable of operating continuously without degrading the SCSI target device's operation or reliability beyond manufacturer accepted limits; or
- b) If no reference temperature is returned, then:
  - A) The log parameter with parameter code 0001h may not be included in the log page; or
  - B) The parameter value in the reference temperature log parameter (parameter code 0001h) may be set to FFh.

The reference temperature may change for vendor specific reasons.

The FORMAT AND LINKING field for log parameter 0001h (i.e., the Reference Temperature parameter) in the Temperature log page shall be set to 11b, indicating that the parameter is a binary format list parameter. The values of the bits and fields in the parameter control byte for binary format list parameters are described in 7.2.1.2.2.3.

## 7.3 Medium auxiliary memory attributes

### 7.3.1 Attribute format

Each medium auxiliary memory attribute shall be communicated between the application client and device server in the format shown in table 326. This format shall be used in the parameter data for the WRITE ATTRIBUTE command (see 6.38) and the READ ATTRIBUTE command (see 6.15). The attribute format in this standard implies nothing about the physical representation of an attribute in the medium auxiliary memory.

**Table 326 — MAM ATTRIBUTE format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____							
1	ATTRIBUTE IDENTIFIER _____ (LSB)							
2	READ ONLY	Reserved					FORMAT	
3	(MSB) _____							
4	ATTRIBUTE LENGTH (n-4) _____ (LSB)							
5	_____							
n	ATTRIBUTE VALUE _____							

The ATTRIBUTE IDENTIFIER field contains a code value identifying the attribute (see 7.3.2).

The READ ONLY bit indicates whether the attribute is in the read only state (see 5.11). If the READ ONLY bit is set to one, the attribute is in the read only state. If the READ ONLY bit is set to zero, the attribute is in the read/write state.

The FORMAT field (see table 327) specifies the format of the data in the ATTRIBUTE VALUE field.

**Table 327 — MAM attribute FORMAT field**

Format	Name	Description
00b	BINARY	The ATTRIBUTE VALUE field contains binary data.
01b	ASCII	The ATTRIBUTE VALUE field contains left-aligned ASCII data (see 4.4.1).
10b	TEXT	The attribute contains textual data. The character set is as described in the TEXT LOCALIZATION IDENTIFIER attribute (see 7.3.2.4.6).
11b		Reserved

The ATTRIBUTE LENGTH field specifies the length in bytes of the ATTRIBUTE VALUE field.

The ATTRIBUTE VALUE field contains the current value, for the READ ATTRIBUTE command (see 6.15), or intended value, for the WRITE ATTRIBUTE command (see 6.38), of the attribute.

## 7.3.2 Attribute identifier values

### 7.3.2.1 Attribute identifier values overview

The values in the ATTRIBUTE IDENTIFIER field (see 7.3.1) are assigned according to the attribute type (see 5.11) and whether the attribute is standard or vendor specific (see table 328).

**Table 328 — MAM attribute identifier range assignments**

Attribute Identifiers	Attribute Type	Standardized	Subclause
0000h to 03FFh	Device	Yes	7.3.2.2
0400h to 07FFh	Medium	Yes	7.3.2.3
0800h to 0BFFh	Host	Yes	7.3.2.4
0C00h to 0FFFh	Device	Vendor specific	
1000h to 13FFh	Medium	Vendor specific	
1400h to 17FFh	Host	Vendor specific	
1800h to FFFFh	Reserved		

Device servers may accept and process a WRITE ATTRIBUTE command containing standardized host type attribute identifier values (i.e., 0800h-0BFFh) or vendor specific host type attribute identifier values (i.e., 1400h-17FFh). Standardized host type attribute identifier values may be checked as described in 7.3.2.4.

### 7.3.2.2 Device type attributes

Device type attributes (see table 329) shall be maintained and updated by the device server when the medium and associated medium auxiliary memory are present. All supported medium type attributes shall have a status of read only (see 5.11).

**Table 329 — Device type attributes**

Attribute Identifier	Name	Attribute Length (in bytes)	Format	Subclause
0000h	REMAINING CAPACITY IN PARTITION	8	BINARY	7.3.2.2.1
0001h	MAXIMUM CAPACITY IN PARTITION	8	BINARY	7.3.2.2.1
0002h	Restricted			
0003h	LOAD COUNT	8	BINARY	7.3.2.2.2
0004h	MAM SPACE REMAINING	8	BINARY	7.3.2.2.3
0005h to 0006h	Restricted			
0007h	INITIALIZATION COUNT	2	BINARY	7.3.2.2.4
0008h	VOLUME IDENTIFIER	32	ASCII	7.3.2.2.5
0009h to 0209h	Reserved			
020Ah	DEVICE VENDOR/SERIAL NUMBER AT LAST LOAD	40	ASCII	7.3.2.2.6
020Bh	DEVICE VENDOR/SERIAL NUMBER AT LOAD-1	40	ASCII	7.3.2.2.6
020Ch	DEVICE VENDOR/SERIAL NUMBER AT LOAD-2	40	ASCII	7.3.2.2.6
020Dh	DEVICE VENDOR/SERIAL NUMBER AT LOAD-3	40	ASCII	7.3.2.2.6
020Eh to 021Fh	Reserved			
0220h	TOTAL MBYTES WRITTEN IN MEDIUM LIFE	8	BINARY	7.3.2.2.7
0221h	TOTAL MBYTES READ IN MEDIUM LIFE	8	BINARY	7.3.2.2.7
0222h	TOTAL MBYTES WRITTEN IN CURRENT/LAST LOAD	8	BINARY	7.3.2.2.8
0223h	TOTAL MBYTES READ IN CURRENT/LAST LOAD	8	BINARY	7.3.2.2.8
0224h	LOGICAL POSITION OF FIRST ENCRYPTED BLOCK	8	BINARY	7.3.2.2.9
0225h to 033Fh	Reserved			
0340h	MEDIUM USAGE HISTORY	90	BINARY	7.3.2.2.10
0341h	PARTITION USAGE HISTORY	60	BINARY	7.3.2.2.11
0342h to 03FFh	Reserved			

**7.3.2.2.1 REMAINING CAPACITY IN PARTITION and MAXIMUM CAPACITY IN PARTITION:** Are native capacities (i.e., assuming no data compression for the specified medium partition). These values are expressed in increments of 1 048 576 bytes (e.g., a value of one means 1 048 576 bytes and a value of two means 2 097 152 bytes).

**7.3.2.2.2 LOAD COUNT:** Indicates how many times this medium has been fully loaded. This attribute should not be reset to zero by any action of the device server. The load counter is a saturating counter.

**7.3.2.2.3 MAM SPACE REMAINING:** Indicates the space currently available in the medium auxiliary memory. The total medium auxiliary memory capacity is reported in the MAM CAPACITY attribute (see 7.3.2.3.4).

NOTE 49 - It may not always be possible to utilize all of the available space in a given medium auxiliary memory implementation. Depending on the internal organization of the memory and the software that controls it, fragmentation issues may mean that certain attribute sizes may not be fully accommodated as the medium auxiliary memory nears its maximum capacity.

**7.3.2.2.4 INITIALIZATION COUNT:** Indicates the number of times that a device server has logically formatted the medium. This value is cumulative over the life of the medium and shall not be reset to zero. The initialization counter is a saturating counter.

**7.3.2.2.5 VOLUME IDENTIFIER:** Indicates the volume identifier (see SMC-3) of the medium. If the device server supports this attribute but does not have access to the volume identifier, it shall report this attribute with an attribute length value of zero.

**7.3.2.2.6 DEVICE VENDOR/SERIAL NUMBER AT LAST LOAD, DEVICE VENDOR/SERIAL NUMBER AT LOAD -1, DEVICE VENDOR/SERIAL NUMBER AT LOAD -2 and DEVICE VENDOR/SERIAL NUMBER AT LOAD -3:**

Give a history of the last four device servers in which the medium has been loaded. The format of the attributes is shown in table 330.

**Table 330 — DEVICE VENDOR/SERIAL NUMBER attribute format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____							
7	T10 VENDOR IDENTIFICATION _____ (LSB)							
8	(MSB) _____							
39	PRODUCT SERIAL NUMBER _____ (LSB)							

The T10 VENDOR IDENTIFICATION field shall be the same value returned in the Standard INQUIRY data (see 6.4.2).

The PRODUCT SERIAL NUMBER field contains ASCII data (see 4.4.1) that is a vendor specific serial number. If the product serial number is not available, the PRODUCT SERIAL NUMBER field shall contain ASCII spaces (20h).

**7.3.2.2.7 TOTAL MBYTES WRITTEN IN MEDIUM LIFE and TOTAL MBYTES READ IN MEDIUM LIFE:** Indicate the total number of data bytes that are transferred to or from the medium, after any data compression has been applied, over the entire medium life. These values are cumulative and shall not be reset to zero. These values are expressed in increments of 1 048 576 bytes (e.g., a value of one means 1 048 576 bytes and a value of two means 2 097 152 bytes).

**7.3.2.2.8 TOTAL MBYTES WRITTEN IN CURRENT/LAST LOAD and TOTAL MBYTES READ IN CURRENT/ LAST LOAD:** Indicate the total number of data bytes that are transferred to or from the medium, after any data compression has been applied, during the current load if the medium is currently loaded, or the last load if the medium is currently unloaded. The device server should reset these attributes to zero when the medium is loaded. These values are expressed in increments of 1 048 576 bytes (e.g., a value of one means 1 048 576 bytes and a value of two means 2 097 152 bytes).

**7.3.2.2.9 LOGICAL POSITION OF FIRST ENCRYPTED BLOCK:** Indicates the address of the first logical block on the medium that contains encrypted data.

**7.3.2.2.10 MEDIUM USAGE HISTORY:** Provides saturating counters (see table 331) for the entire medium. The value in each field is the sum for all partitions. If a field is not used, it should be set to zero.

**Table 331 — MEDIUM USAGE HISTORY attribute format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)	CURRENT AMOUNT OF DATA WRITTEN						(LSB)
5								
6	(MSB)	CURRENT WRITE RETRIES COUNT						(LSB)
11								
12	(MSB)	CURRENT AMOUNT OF DATA READ						(LSB)
17								
18	(MSB)	CURRENT READ RETRIES COUNT						(LSB)
23								
24	(MSB)	PREVIOUS AMOUNT OF DATA WRITTEN						(LSB)
29								
30	(MSB)	PREVIOUS WRITE RETRIES COUNT						(LSB)
35								
36	(MSB)	PREVIOUS AMOUNT OF DATA READ						(LSB)
41								
42	(MSB)	PREVIOUS READ RETRIES COUNT						(LSB)
47								
48	(MSB)	TOTAL AMOUNT OF DATA WRITTEN						(LSB)
53								
54	(MSB)	TOTAL WRITE RETRIES COUNT						(LSB)
59								
60	(MSB)	TOTAL AMOUNT OF DATA READ						(LSB)
65								
66	(MSB)	TOTAL READ RETRIES COUNT						(LSB)
71								
72	(MSB)	LOAD COUNT						(LSB)
77								
78	(MSB)	TOTAL CHANGE PARTITION COUNT						(LSB)
83								
84	(MSB)	TOTAL PARTITION INITIALIZE COUNT						(LSB)
89								

The CURRENT AMOUNT OF DATA WRITTEN field indicates the amount of data written to the medium during this load of the medium. This value is expressed in mebibytes (see 3.6.4).

The CURRENT WRITE RETRIES COUNT field indicates the total number of times a write retry occurred during this load of the medium.<sup>1)</sup>

The CURRENT AMOUNT OF DATA READ field indicates the amount of data read from the medium during this load of the medium. This value is expressed in mebibytes (see 3.6.4).

The CURRENT READ RETRIES COUNT field indicates the number of times a read retry occurred during this load of the medium.<sup>1)</sup>

The PREVIOUS AMOUNT OF DATA WRITTEN field indicates the amount of data written to the medium during the previous load of the medium. This value is expressed in mebibytes (see 3.6.4).

The PREVIOUS WRITE RETRIES COUNT field indicates the total number of times a write retry occurred during the previous load of the medium.<sup>1)</sup>

The PREVIOUS AMOUNT OF DATA READ field indicates the amount of data read from the medium during the previous load of the medium. This value is expressed in mebibytes (see 3.6.4).

The PREVIOUS READ RETRIES COUNT field indicates the number of times a read retry occurred during the previous load of the medium.<sup>1)</sup>

The TOTAL AMOUNT OF DATA WRITTEN field indicates the amount of data written to the medium since the last medium format. This value is expressed in mebibytes (see 3.6.4).

The TOTAL WRITE RETRIES COUNT field indicates the total number of times a write retry occurred since the last medium format.<sup>1)</sup>

The TOTAL AMOUNT OF DATA READ field indicates the amount of data read from the medium since the last medium format. This value is expressed in mebibytes (see 3.6.4).

The TOTAL READ RETRIES COUNT field indicates the number of times a read retry occurred since the last medium format.<sup>1)</sup>

The LOAD COUNT field indicates the number of loads since the last medium format. This count accumulates over the life of the medium but it is reset to zero after a medium format.

The TOTAL CHANGE PARTITION COUNT field indicates the number of times that switches between partitions have been performed on the medium. This count accumulates over the life of the medium but it is reset to zero after a medium format.

The TOTAL PARTITION INITIALIZE COUNT field indicates number of times that any of the partitions on the medium have been erased. This count accumulates over the life of the medium but it is reset to zero after a medium format.

---

1) The definition of one retry as counted by this attribute field is not part of this standard. This count should not be used to compare products because the products may define errors differently.



**7.3.2.2.11 PARTITION USAGE HISTORY:** Provides saturating counters (see table 332) for the partition specified by the PARTITION NUMBER field in the CDB. If a field is not used, it should be set to zero.

**Table 332 — PARTITION USAGE HISTORY attribute format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)	CURRENT AMOUNT OF DATA WRITTEN						(LSB)
3								
4	(MSB)	CURRENT WRITE RETRIES COUNT						(LSB)
7								
8	(MSB)	CURRENT AMOUNT OF DATA READ						(LSB)
11								
12	(MSB)	CURRENT READ RETRIES COUNT						(LSB)
15								
16	(MSB)	PREVIOUS AMOUNT OF DATA WRITTEN						(LSB)
19								
20	(MSB)	PREVIOUS WRITE RETRIES COUNT						(LSB)
23								
24	(MSB)	PREVIOUS AMOUNT OF DATA READ						(LSB)
27								
28	(MSB)	PREVIOUS READ RETRIES COUNT						(LSB)
31								
32	(MSB)	TOTAL AMOUNT OF DATA WRITTEN						(LSB)
35								
36	(MSB)	TOTAL WRITE RETRIES COUNT						(LSB)
39								
40	(MSB)	TOTAL AMOUNT OF DATA READ						(LSB)
43								
44	(MSB)	TOTAL READ RETRIES COUNT						(LSB)
47								
48	(MSB)	LOAD COUNT						(LSB)
51								
52	(MSB)	CHANGE PARTITION COUNT						(LSB)
55								
56	(MSB)	PARTITION INITIALIZE COUNT						(LSB)
59								

The CURRENT AMOUNT OF DATA WRITTEN field indicates the amount of data written to the medium in the partition specified by the PARTITION NUMBER field in the CDB during this load of the medium. This value is expressed in mebibytes (see 3.6.4).

The CURRENT WRITE RETRIES COUNT field indicates the total number of times a write retry occurred in the partition specified by the PARTITION NUMBER field in the CDB during this load of the medium.<sup>2)</sup>

The CURRENT AMOUNT OF DATA READ field indicates the amount of data read from the medium in the partition specified by the PARTITION NUMBER field in the CDB during this load of the medium. This value is expressed in mebibytes (see 3.6.4).

The CURRENT READ RETRIES COUNT field indicates the number of times a read retry occurred in the partition specified by the PARTITION NUMBER field in the CDB during this load of the medium.<sup>2)</sup>

The PREVIOUS AMOUNT OF DATA WRITTEN field indicates the amount of data written to the medium in the partition specified by the PARTITION NUMBER field in the CDB during the previous load of the medium. This value is expressed in mebibytes (see 3.6.4).

The PREVIOUS WRITE RETRIES COUNT field indicates the total number of times a write retry occurred in the partition specified by the PARTITION NUMBER field in the CDB during the previous load of the medium.<sup>2)</sup>

The PREVIOUS AMOUNT OF DATA READ field indicates the amount of data read from the medium in the partition specified by the PARTITION NUMBER field in the CDB during the previous load of the medium. This value is expressed in mebibytes (see 3.6.4).

The PREVIOUS READ RETRIES COUNT field indicates the number of times a read retry occurred in the partition specified by the PARTITION NUMBER field in the CDB during the previous load of the medium.<sup>2)</sup>

The TOTAL AMOUNT OF DATA WRITTEN field indicates the amount of data written to the medium in the partition specified by the PARTITION NUMBER field in the CDB since the last medium format. This value is expressed in mebibytes (see 3.6.4).

The TOTAL WRITE RETRIES COUNT field indicates the total number of times a write retry occurred in the partition specified by the PARTITION NUMBER field in the CDB since the last medium format.<sup>2)</sup>

The TOTAL AMOUNT OF DATA READ field indicates the amount of data read from the medium in the partition specified by the PARTITION NUMBER field in the CDB since the last medium format. This value is expressed in mebibytes (see 3.6.4).

The TOTAL READ RETRIES COUNT field indicates the number of times a read retry occurred in the partition specified by the PARTITION NUMBER field in the CDB since the last medium format.<sup>2)</sup>

The LOAD COUNT field indicates the number of loads in the partition specified by the PARTITION NUMBER field in the CDB since the last medium format. This count accumulates over the life of the medium but it is reset to zero after a medium format.

The TOTAL CHANGE PARTITION COUNT field indicates the number of times that switches to the partition specified by the PARTITION NUMBER field in the CDB have been performed on the medium. This count accumulates over the life of the medium but it is reset to zero after a medium format.

The TOTAL PARTITION INITIALIZE COUNT field indicates number of times that the partition specified by the PARTITION NUMBER field in the CDB has been initialized. This count accumulates over the life of the medium but it is reset to zero after a medium format.

---

2) The definition of one retry as counted by this attribute field is not part of this standard. This count should not be used to compare products because the products may define errors differently.

### 7.3.2.3 Medium type attributes

Medium type attributes (see table 333) are stored in the medium auxiliary memory by the manufacturer. The device server shall not alter medium type attributes. All supported medium type attributes shall have a status of read only (see 5.11).

**Table 333 — Medium type attributes**

Attribute Identifier	Name	Attribute Length (in bytes)	Format	Subclause
0400h	MEDIUM MANUFACTURER	8	ASCII	7.3.2.3.1
0401h	MEDIUM SERIAL NUMBER	32	ASCII	7.3.2.3.2
0402h to 0405h	Restricted			
0406h	MEDIUM MANUFACTURE DATE	8	ASCII	7.3.2.3.3
0407h	MAM CAPACITY	8	BINARY	7.3.2.3.4
0408h	MEDIUM TYPE	1	BINARY	7.3.2.3.5
0409h	MEDIUM TYPE INFORMATION	2	BINARY	7.3.2.3.5
040Ah	NUMERIC MEDIUM SERIAL NUMBER	unspecified	unspecified	7.3.2.3.6
040Bh to 07FFh	Reserved			

**7.3.2.3.1 MEDIUM MANUFACTURER:** Contains eight bytes of left-aligned ASCII data (see 4.4.1) identifying the vendor of the media. The medium manufacturer shall be a T10 vendor identification assigned by INCITS. A list of assigned T10 vendor identifications is in Annex E and on the T10 web site (<http://www.T10.org>).

NOTE 50 - The T10 web site (<http://www.t10.org>) provides a convenient means to request an identification code.

**7.3.2.3.2 MEDIUM SERIAL NUMBER:** Contains the manufacturer's serial number for the medium.

**7.3.2.3.3 MEDIUM MANUFACTURE DATE:** Contains the date of manufacture of the medium. The format is YYYYMMDD (i.e., four numeric ASCII characters for the year followed by two numeric ASCII characters for the month followed by two numeric ASCII characters for the day with no intervening spaces).

**7.3.2.3.4 MAM CAPACITY:** Is the total capacity of the medium auxiliary memory, in bytes, at manufacture time. It does not indicate the available space of an unused medium auxiliary memory because some of the medium auxiliary memory space may be reserved for device-specific use making it inaccessible to the application client.

**7.3.2.3.5 MEDIUM TYPE and MEDIUM TYPE INFORMATION:** Give information about non-data media and other types of media. The MEDIUM TYPE INFORMATION attribute is interpreted according to the type of medium indicated by the MEDIUM TYPE (see table 334).

**Table 334 — MEDIUM TYPE and MEDIUM TYPE INFORMATION attributes**

MEDIUM TYPE	Description	MEDIUM TYPE INFORMATION
00h	Data medium	Reserved
01h	Cleaning medium	Maximum number of cleaning cycles permitted
02h to 7Fh	Reserved	Reserved
80h	Write-once medium	Reserved
81h to FFh	Reserved	Reserved

**7.3.2.3.6 NUMERIC MEDIUM SERIAL NUMBER:** Contains the manufacturer's serial number for the medium in a vendor specific format.

#### 7.3.2.4 Host type attributes

Application clients may use the WRITE ATTRIBUTE and READ ATTRIBUTE commands to maintain the attributes shown in table 335. All existent host type attributes shall have a status of read/write (see 5.11).

**Table 335 — Host type attributes**

Attribute Identifier	Name	Attribute Length (in bytes)	Format	Subclause
0800h	APPLICATION VENDOR	8	ASCII	7.3.2.4.1
0801h	APPLICATION NAME	32	ASCII	7.3.2.4.2
0802h	APPLICATION VERSION	8	ASCII	7.3.2.4.3
0803h	USER MEDIUM TEXT LABEL	160	TEXT	7.3.2.4.4
0804h	DATE AND TIME LAST WRITTEN	12	ASCII	7.3.2.4.5
0805h	TEXT LOCALIZATION IDENTIFIER	1	BINARY	7.3.2.4.6
0806h	BARCODE	32	ASCII	7.3.2.4.7
0807h	OWNING HOST TEXTUAL NAME	80	TEXT	7.3.2.4.8
0808h	MEDIA POOL	160	TEXT	7.3.2.4.9
0809h	PARTITION USER TEXT LABEL	16	ASCII	7.3.2.4.10
080Ah	LOAD/UNLOAD AT PARTITION	1	BINARY	7.3.2.4.11
080Bh to BFFh	Reserved			

**7.3.2.4.1 APPLICATION VENDOR:** Contains eight bytes of left-aligned ASCII data (see 4.4.1) identifying the manufacturer of the application client (e.g., class driver or backup program) that last sent a WRITE ATTRIBUTE command to the device server while this medium auxiliary memory was accessible. The application vendor shall be a T10 vendor identification assigned by INCITS. A list of assigned T10 vendor identifications is in Annex E and on the T10 web site (<http://www.T10.org>).

NOTE 51 - The T10 web site (<http://www.t10.org>) provides a convenient means to request an identification code.

**7.3.2.4.2 APPLICATION NAME:** Contains the name of the application client.

**7.3.2.4.3 APPLICATION VERSION:** Contains the version of the application client.

**7.3.2.4.4 USER MEDIUM TEXT LABEL:** Is the user level identifier for the medium.

**7.3.2.4.5 DATE & TIME LAST WRITTEN:** Contains when the application client last wrote to the medium auxiliary memory. The format is YYYYMMDDHHMM (i.e., four numeric ASCII characters for the year followed by two numeric ASCII characters for the month followed by two numeric ASCII characters for the day followed by two numeric ASCII characters between 00 and 24 for the hour followed by two numeric ASCII characters for the minute with no intervening spaces).

**7.3.2.4.6 TEXT LOCALIZATION IDENTIFIER:** Defines the character set (see table 336) used for attributes with a TEXT format (see 7.3.1).

**Table 336 — TEXT LOCALIZATION IDENTIFIER attribute values**

Value	Meaning
00h	No code specified (ASCII)
01h	ISO/IEC 8859-1 (Europe, Latin America)
02h	ISO/IEC 8859-2 (Eastern Europe)
03h	ISO/IEC 8859-3 (SE Europe/miscellaneous)
04h	ISO/IEC 8859-4 (Scandinavia/Baltic)
05h	ISO/IEC 8859-5 (Cyrillic)
06h	ISO/IEC 8859-6 (Arabic)
07h	ISO/IEC 8859-7 (Greek)
08h	ISO/IEC 8859-8 (Hebrew)
09h	ISO/IEC 8859-9 (Latin 5)
0Ah	ISO/IEC 8859-10 (Latin 6)
0Bh to 7Fh	Reserved
80h	ISO/IEC 10646-1 (UCS-2BE)
81h	ISO/IEC 10646-1 (UTF-8)
82h to FFh	Reserved

**7.3.2.4.7 BARCODE:** Is contents of a barcode associated with the medium in the medium auxiliary memory.

**7.3.2.4.8 OWNING HOST TEXTUAL NAME:** Indicates the host from which that USER MEDIUM TEXT LABEL (see 7.3.2.4.4) originates.

**7.3.2.4.9 MEDIA POOL:** Indicates the media pool to which this medium belongs.

**7.3.2.4.10 PARTITION USER TEXT LABEL:** Is a user level identifier for the partition specified by the PARTITION NUMBER field in the CDB.

**7.3.2.4.11 LOAD/UNLOAD AT PARTITION:** Indicates whether the media is capable of being loaded or unloaded at the partition specified by the PARTITION NUMBER field in the CDB. If loads and unloads are enabled for the specified partition, the value of this attribute shall be one. If loads and unloads are not enabled for the specified partition, the value of this attribute shall be zero. All attribute values other than zero and one are reserved. If LOAD/UNLOAD AT PARTITION is disabled, then loads and unloads are performed at the beginning of the media instead of at the specified partition. If this attribute is in the nonexistent state (see 5.11), then the default action shall be to load and unload at the beginning of media.

## 7.4 Mode parameters

### 7.4.1 Mode parameters overview

This subclause describes the mode parameter headers, block descriptors, and mode pages used with MODE SELECT command (see 6.9 and 6.10) and MODE SENSE command (see 6.11 and 6.12) that are applicable to all SCSI devices. Subpages are identical to mode pages except that they include a SUBPAGE CODE field that further differentiates the mode page contents. Mode pages specific to each device type are described in the command standard (see 3.1.27) that applies to that device type.

### 7.4.2 Mode parameter list format

The mode parameter list shown in table 337 contains a header, followed by zero or more block descriptors, followed by zero or more variable-length mode pages. Parameter lists are defined for each device type.

**Table 337 — Mode parameter list**

Bit Byte	7	6	5	4	3	2	1	0
	Mode parameter header							
	Block descriptor(s)							
	Mode page(s) or vendor specific (e.g., page code set to zero)							

### 7.4.3 Mode parameter header formats

The mode parameter header that is used by the MODE SELECT(6) command (see 6.9) and the MODE SENSE(6) command (see 6.11) is defined in table 338.

**Table 338 — Mode parameter header(6)**

Bit Byte	7	6	5	4	3	2	1	0
0	MODE DATA LENGTH							
1	MEDIUM TYPE							
2	DEVICE-SPECIFIC PARAMETER							
3	BLOCK DESCRIPTOR LENGTH							

The mode parameter header that is used by the MODE SELECT(10) command (see 6.10) and the MODE SENSE(10) command (see 6.12) is defined in table 339.

**Table 339 — Mode parameter header(10)**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____							
1	MODE DATA LENGTH _____ (LSB)							
2	MEDIUM TYPE							
3	DEVICE-SPECIFIC PARAMETER							
4	Reserved							LONGLBA
5	Reserved							
6	(MSB) _____							
7	BLOCK DESCRIPTOR LENGTH _____ (LSB)							

When using the MODE SENSE command, the MODE DATA LENGTH field indicates the length in bytes of the following data that is available to be transferred. The mode data length does not include the number of bytes in the MODE DATA LENGTH field. When using the MODE SELECT command, this field is reserved.

NOTE 52 - Logical units that support more than 256 bytes of block descriptors and mode pages may need to implement ten-byte mode commands. The mode data length field in the six-byte CDB header limits the returned data to 256 bytes.

The contents of the MEDIUM TYPE field are unique for each device type. Refer to the mode parameters subclause of the specific device type command standard (see 3.1.27) for definition of these values. Some device types reserve this field.

The DEVICE-SPECIFIC PARAMETER field is unique for each device type. Refer to the mode parameters subclause of the specific device type command standard (see 3.1.27) for definition of this field. Some device types reserve all or part of this field.

If the Long LBA (LONGLBA) bit is set to zero, the mode parameter block descriptor(s), if any, are each eight bytes long and have the format described in 7.4.4.1. If the LONGLBA bit is set to one, the mode parameter block descriptor(s), if any, are each sixteen bytes long and have a format described in a command standard (see 3.1.27).

The BLOCK DESCRIPTOR LENGTH field contains the length in bytes of all the block descriptors. It is equal to the number of block descriptors times eight if the LONGLBA bit is set to zero or times sixteen if the LONGLBA bit is set to one, and does not include mode pages or vendor specific parameters (e.g., page code set to zero), if any, that may follow the last block descriptor. A block descriptor length of zero indicates that no block descriptors are included in the mode parameter list. This condition shall not be considered an error.

## 7.4.4 Mode parameter block descriptor formats

### 7.4.4.1 General block descriptor format

When the LONGLBA bit is set to zero (see 7.4.3), the mode parameter block descriptor format for all device types except direct access block devices (see SBC-2) is shown in table 340.

**Table 340 — General mode parameter block descriptor**

Bit Byte	7	6	5	4	3	2	1	0
0	DENSITY CODE							
1	(MSB)							
2	NUMBER OF BLOCKS							
3								(LSB)
4	Reserved							
5	(MSB)							
6	BLOCK LENGTH							
7								(LSB)

Block descriptors specify some of the medium characteristics for all or part of a logical unit. Support for block descriptors is optional. Each block descriptor contains a DENSITY CODE field, a NUMBER OF BLOCKS field, and a BLOCK LENGTH field. Block descriptor values are always current (i.e., saving is not supported). Whenever any block descriptor values are changed, the device server shall establish a unit attention condition (see SAM-4) for the initiator port associated with every I\_T nexus except the I\_T nexus on which the MODE SELECT command (see 6.9) was received, with the additional sense code set to MODE PARAMETERS CHANGED. Command standards (see 3.1.27) may place additional requirements on the general mode parameter block descriptor. Requirements in the command standards that conflict with requirements defined in this subclause shall take precedence over the requirements defined in this subclause.

The DENSITY CODE field is unique for each device type. Refer to the mode parameters subclause of the specific device type command standard (see 3.1.27) for definition of this field. Some device types reserve all or part of this field.

The NUMBER OF BLOCKS field specifies the number of logical blocks on the medium to which the DENSITY CODE field and BLOCK LENGTH field apply. A value of zero indicates that all of the remaining logical blocks of the logical unit shall have the medium characteristics specified.

If the number of logical blocks on the medium exceeds the maximum value that may be specified in the NUMBER OF BLOCKS field, a value of FFFFFFFh indicates that all of the remaining logical blocks of the logical unit shall have the medium characteristics specified.

#### NOTES

- 53 There may be implicit association between parameters defined in the mode pages and block descriptors. In this case, the device server may change parameters not explicitly sent with the MODE SELECT command. A subsequent MODE SENSE command may be used to detect these changes.
- 54 The number of remaining logical blocks may be unknown for some device types.

The BLOCK LENGTH field specifies the length in bytes of each logical block described by the block descriptor. For sequential-access devices, a block length of zero indicates that the logical block size written to the medium is specified by the TRANSFER LENGTH field in the CDB (see SSC-2).



### 7.4.5 Mode page and subpage formats and page codes

The page\_0 mode page format is defined in table 341.

**Table 341 — Page\_0 mode page format**

Bit Byte	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	PAGE CODE					
1	PAGE LENGTH (n-1)							
2	Mode parameters							
n								

The sub\_page mode page format is defined in table 342.

**Table 342 — Sub\_page mode page format**

Bit Byte	7	6	5	4	3	2	1	0
0	PS	SPF (1b)	PAGE CODE					
1	SUBPAGE CODE							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
4	Mode parameters							
n								

Each mode page contains a PS bit, an SPF bit, a PAGE CODE field, a PAGE LENGTH field, and a set of mode parameters. The page codes are defined in this subclause and in the mode parameter subclauses in the command standard (see 3.1.27) for the specific device type. Each mode page with a SPF bit set to one contains a SUBPAGE CODE field.

A SubPage Format (SPF) bit set to zero indicates that the page\_0 mode page format is being used. A SPF bit set to one indicates that the sub\_page mode page format is being used.

When using the MODE SENSE command, a parameters saveable (PS) bit set to one indicates that the mode page may be saved by the logical unit in a nonvolatile, vendor specific location. A PS bit set to zero indicates that the device server is not able to save the supported parameters. When using the MODE SELECT command, the PS bit is reserved.

The PAGE CODE and SUBPAGE CODE fields identify the format and parameters defined for that mode page. Some page codes are defined as applying to all device types and other page codes are defined for the specific device type. The page codes that apply to a specific device type are defined in the command standard (see 3.1.27) for that device type. The applicability of each subpage code matches that of the page code with which it is associated.

When using the MODE SENSE command, if page code 00h (vendor specific mode page) is implemented, the device server shall return that mode page last in response to a request to return all mode pages (page code 3Fh). When using the MODE SELECT command, this mode page should be sent last.

The PAGE LENGTH field specifies the length in bytes of the mode parameters that follow. If the application client does not set this value to the value that is returned for the mode page by the MODE SENSE command, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST. The logical unit may implement a mode page that is less than the full mode page length defined, provided no field is truncated and the PAGE LENGTH field correctly specifies the actual length implemented.

The mode parameters for each mode page are defined in the following subclauses, or in the mode parameters subclause in the command standard (see 3.1.27) for the specific device type. Mode parameters not implemented by the logical unit shall be set to zero.

Table 343 defines the mode pages that are applicable to all device types that implement the MODE SELECT and MODE SENSE commands.

**Table 343 — Mode page codes and subpage codes**

Page code	Subpage code	Mode Page Name	Reference
0Ah	00h	Control	7.4.6
0Ah	01h	Control Extension	7.4.7
02h	00h	Disconnect-Reconnect	7.4.8
15h	00h	Extended	7.4.9
16h	00h	Extended Device-Type Specific	7.4.10
1Ch	00h	Informational Exceptions Control	7.4.11
09h	00h	obsolete	3.3.7
1Ah	00h	Power Condition	7.4.12
18h	00h	Protocol Specific Logical Unit	7.4.13
18h	01h to FEh	(See specific SCSI transport protocol standard)	7.4.14
19h	00h	Protocol Specific Port	
19h	01h to FEh	(See specific SCSI transport protocol standard)	
01h	00h to FEh	(See command standard for specific device type)	
03h to 08h	00h to FEh	(See command standard for specific device type)	
0Ah	F0h to FEh	(See command standard for specific device type)	
0Bh to 14h	00h to FEh	(See command standard for specific device type)	
1Ah	F0h to FEh	(See command standard for specific device type)	
1Bh	00h to FEh	(See command standard for specific device type)	
1Dh to 1Fh	00h to FEh	(See command standard for specific device type)	
20h to 3Eh	00h to FEh	(See command standard for specific device type)	
00h	vendor specific	Vendor specific (does not require page format)	
3Fh	00h	Return all pages <sup>a</sup>	
3Fh	FFh	Return all pages and subpages <sup>a</sup>	
00h to 3Eh	FFh	Return all subpages <sup>a</sup>	
All page code and subpage code combinations not shown in this table are reserved. Annex D contains a listing of mode page and subpage codes in numeric order.			
<sup>a</sup> Valid only for the MODE SENSE command			

### 7.4.6 Control mode page

The Control mode page (see table 344) provides controls over SCSI features that are applicable to all device types (e.g., task set management and error logging). If a field in this mode page is changed while there is a task already in the task set, it is vendor specific whether the old or new value of the field applies to that task. The mode page policy (see 6.9) for this mode page shall be shared, or per I\_T nexus.

**Table 344 — Control mode page**

Bit Byte	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	PAGE CODE (0Ah)					
1	PAGE LENGTH (0Ah)							
2	TST			TMF_ONLY	Reserved	D_SENSE	GLTSD	RLEC
3	QUEUE ALGORITHM MODIFIER				Reserved	QERR		Obsolete
4	VS	RAC	UA_INTLCK_CTRL		SWP	Obsolete		
5	ATO	TAS	Reserved			AUTOLOAD MODE		
6	Obsolete							
7								
8	(MSB)	BUSY TIMEOUT PERIOD						
9								(LSB)
10	(MSB)	EXTENDED SELF-TEST COMPLETION TIME						
11								(LSB)

The PS bit, SPF bit, PAGE CODE field, and PAGE LENGTH field are described in 7.4.5.

A task set type (TST) field (see table 345) specifies the type of task set in the logical unit.

**Table 345 — Task set type (TST) field**

Code	Description
000b	The logical unit maintains one task set for all I_T nexuses
001b	The logical unit maintains separate task sets for each I_T nexus
010b to 111b	Reserved

Regardless of the mode page policy (see 6.9) for the Control mode page, the shared mode page policy shall be applied to the TST field. If the most recent MODE SELECT changes the setting of this field, then the device server shall establish a unit attention condition (see SAM-4) for the initiator port associated with every I\_T nexus except the I\_T nexus on which the MODE SELECT command was received, with the additional sense code set to MODE PARAMETERS CHANGED.

The allow task management functions only (TMF\_ONLY) bit set to zero specifies that the device server shall process tasks with the ACA task attribute received on the faulted I\_T nexus (see 3.1.53) when an ACA condition has been established (see SAM-4). A TMF\_ONLY bit set to one specifies that the device server shall complete all commands received on the faulted I\_T nexus with an ACA ACTIVE status when an ACA condition has been established.

A descriptor format sense data (D\_SENSE) bit set to zero specifies that the device server shall return fixed format sense data (see 4.5.3) when returning sense data in the same I\_T\_L\_Q nexus transaction (see 3.1.62) as a

CHECK CONDITION status. A D\_SENSE bit set to one specifies that the device server shall return descriptor format sense data (see 4.5.2) when returning sense data in the same I\_T\_L\_Q nexus transaction as a CHECK CONDITION status, except as defined in 4.5.1.

A global logging target save disable (GLTSD) bit set to zero specifies that the logical unit implicitly saves, at vendor specific intervals, each log parameter in which the TSD bit (see 7.2) is set to zero. A GLTSD bit set to one specifies that the logical unit shall not implicitly save any log parameters.

A report log exception condition (RLEC) bit set to one specifies that the device server shall report log exception conditions as described in 7.2.1. A RLEC bit set to zero specifies that the device server shall not report log exception conditions.

The QUEUE ALGORITHM MODIFIER field (see table 346) specifies restrictions on the algorithm used for reordering tasks having the SIMPLE task attribute (see SAM-4).

**Table 346 — QUEUE ALGORITHM MODIFIER field**

Code	Description
0h	Restricted reordering
1h	Unrestricted reordering allowed
2h to 7h	Reserved
8h to Fh	Vendor specific

A value of zero in the QUEUE ALGORITHM MODIFIER field specifies that the device server shall order the processing sequence of tasks having the SIMPLE task attribute such that data integrity is maintained for that I\_T nexus (i.e., if the transmission of new SCSI transport protocol requests is halted at any time, the final value of all data observable on the medium shall have exactly the same value as it would have if all the tasks had been given the ORDERED task attribute).

A value of one in the QUEUE ALGORITHM MODIFIER field specifies that the device server may reorder the processing sequence of tasks having the SIMPLE task attribute in any manner. Any data integrity exposures related to task sequence order shall be explicitly handled by the application client through the selection of appropriate commands and task attributes.

The queue error management (QERR) field (see table 347) specifies how the device server shall handle other tasks when one task is terminated with CHECK CONDITION status (see SAM-4). The task set type (see the TST field definition in this subclause) defines which other tasks are affected. If the TST field equals 000b, then all tasks from all I\_T nexuses are affected. If the TST field equals 001b, then only tasks from the same I\_T nexus as the task that is terminated with CHECK CONDITION status are affected.

**Table 347 — Queue error management (QERR) field**

Code	Definition
00b	If an ACA condition is established, the affected tasks in the task set shall resume after the ACA condition is cleared (see SAM-4). Otherwise, all tasks other than the task that received the CHECK CONDITION status shall be processed as if no error occurred.
01b	All the affected tasks in the task set shall be aborted when the CHECK CONDITION status is sent. If the TAS bit is set to zero, the device server shall establish a unit attention condition (see SAM-4) for the initiator port associated with every I_T nexus that had tasks aborted except for the I_T nexus on which the CHECK CONDITION status was returned, with the additional sense code set to COMMANDS CLEARED BY ANOTHER INITIATOR. If the TAS bit is set to one, all affected tasks in the task set for I_T nexuses other than the I_T nexus for which the CHECK CONDITION status was sent shall be completed with TASK ABORTED status and no unit attention shall be established. For the I_T nexus to which the CHECK CONDITION status is sent, no status shall be sent for the tasks that are aborted.
10b	Reserved
11b	Affected tasks in the task set belonging to the I_T nexus on which a CHECK CONDITION status is returned shall be aborted when the status is sent.

A task aborted status (TAS) bit set to zero specifies that aborted tasks shall be terminated by the device server without any response to the application client. A TAS bit set to one specifies that tasks aborted by the actions of an I\_T nexus other than the I\_T nexus on which the command was received shall be completed with TASK ABORTED status (see SAM-4).

The report a check (RAC) bit provides control of reporting long busy conditions or CHECK CONDITION status. A RAC bit set to one specifies that the device server should return CHECK CONDITION status rather than returning BUSY status if the reason for returning the BUSY status may persist for a longer time than that specified by the BUSY TIMEOUT PERIOD field. A RAC bit set to zero specifies that the device server may return BUSY status regardless of the length of time the reason for returning BUSY status may persist.

The unit attention interlocks control (UA\_INTLCK\_CTRL) field (see table 348) controls the clearing of unit attention conditions reported in the same I\_T\_L\_Q nexus transaction (see 3.1.62) as a CHECK CONDITION status and whether returning a status of BUSY, TASK SET FULL or RESERVATION CONFLICT results in the establishment of a unit attention condition (see SAM-4).

**Table 348 — Unit attention interlocks control (UA\_INTLCK\_CTRL) field**

Code	Definition
00b	The logical unit shall clear any unit attention condition reported in the same I_T_L_Q nexus transaction as a CHECK CONDITION status and shall not establish a unit attention condition when a command is completed with BUSY, TASK SET FULL, or RESERVATION CONFLICT status.
01b	Reserved <sup>a</sup>
10b <sup>a</sup>	The logical unit shall not clear any unit attention condition reported in the same I_T_L_Q nexus transaction as a CHECK CONDITION status and shall not establish a unit attention condition when a command is completed with BUSY, TASK SET FULL, or RESERVATION CONFLICT status.
11b <sup>a</sup>	The logical unit shall not clear any unit attention condition reported in the same I_T_L_Q nexus transaction as a CHECK CONDITION status and shall establish a unit attention condition for the initiator port associated with the I_T nexus on which the BUSY, TASK SET FULL, or RESERVATION CONFLICT status is being returned. Depending on the status, the additional sense code shall be set to PREVIOUS BUSY STATUS, PREVIOUS TASK SET FULL STATUS, or PREVIOUS RESERVATION CONFLICT STATUS. Until it is cleared by a REQUEST SENSE command, a unit attention condition shall be established only once for a BUSY, TASK SET FULL, or RESERVATION CONFLICT status regardless to the number of commands completed with one of those status codes.
<sup>a</sup> A REQUEST SENSE command still clears any unit attention condition that it reports.	

A software write protect (SWP) bit set to one specifies that the logical unit shall inhibit writing to the medium after writing all cached or buffered write data, if any. When SWP is one, all commands requiring writes to the medium shall be terminated with CHECK CONDITION status, with the sense key set to DATA PROTECT, and the additional sense code set to WRITE PROTECTED. When SWP is one and the device type's command standard (see 3.1.27) defines a write protect (WP) bit in the DEVICE-SPECIFIC PARAMETER field in the mode parameter header, the WP bit shall be set to one for subsequent MODE SENSE commands. A SWP bit set to zero specifies that the logical unit may allow writing to the medium, depending on other write inhibit mechanisms implemented by the logical unit. When the SWP bit is set to zero, the value of the WP bit, if defined, is device type specific. For a list of commands affected by the SWP bit and details of the WP bit see the command standard for the specific device type.

An application tag owner (ATO) bit set to zero specifies that the device server may modify the contents of the LOGICAL BLOCK APPLICATION TAG field and, depending on the protection type, may modify the contents of the LOGICAL BLOCK REFERENCE TAG field (see SBC-3). If the ATO bit is set to one the device server shall not modify the LOGICAL BLOCK APPLICATION TAG field and, depending on the protection type, shall not modify the contents of the LOGICAL BLOCK REFERENCE TAG field.

The AUTOLOAD MODE field specifies the action to be taken by a removable medium device server when a medium is inserted. For devices other than removable medium devices, this field is reserved. Table 349 shows the usage of the AUTOLOAD MODE field.

**Table 349 — AUTOLOAD MODE field**

Code	Definition
000b	Medium shall be loaded for full access.
001b	Medium shall be loaded for medium auxiliary memory access only.
010b	Medium shall not be loaded.
011b to 111b	Reserved

The BUSY TIMEOUT PERIOD field specifies the maximum time, in 100 milliseconds increments, that the application client allows for the device server to return BUSY status for unanticipated conditions that are not a routine part of commands from the application client. This value may be rounded down as defined in 5.4. A 0000h value in this field is undefined by this standard. An FFFFh value in this field is defined as an unlimited period.

The EXTENDED SELF-TEST COMPLETION TIME field contains advisory data that is the time in seconds that the device server requires to complete an extended self-test when the device server is not interrupted by subsequent commands and no errors occur during processing of the self-test. The application client should expect this time to increase significantly if other commands are sent to the logical unit while a self-test is in progress or if errors occur during the processing of the self-test. Device servers supporting SELF-TEST CODE field values other than 000b for the SEND DIAGNOSTIC command (see 6.32) shall support the EXTENDED SELF-TEST COMPLETION TIME field. The EXTENDED SELF-TEST COMPLETION TIME field is not changeable. A value of FFFFh indicates that the extended self-test takes 65 535 seconds or longer.

Bits 0, 1, and 2 of byte 4 as well as bytes 6 and 7 provide controls for the obsolete asynchronous event reporting feature.

### 7.4.7 Control Extension mode page

The Control Extension mode page (see table 350) is a subpage of the Control mode page (see 7.4.6) and provides controls over SCSI features that are applicable to all device types. The mode page policy (see 6.9) for this mode page shall be shared. If a field in this mode page is changed while there is a task already in the task set, it is vendor specific whether the old or new value of the field applies to that task.

**Table 350 — Control Extension mode page**

Bit Byte	7	6	5	4	3	2	1	0
0	PS	SPF (1b)	PAGE CODE (0Ah)					
1	SUBPAGE CODE (01h)							
2	(MSB)	PAGE LENGTH (1Ch)						(LSB)
3								
4	Reserved					TCMOS	SCSIP	IALUAE
5	Reserved				INITIAL COMMAND PRIORITY			
6	Reserved							
31								

The PS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.4.5.

A SCSI precedence (SCSIP) bit set to one specifies that the timestamp changed using a SET TIMESTAMP command (see 6.36) shall take precedence over methods outside the scope of this standard. A SCSIP bit set to zero specifies that methods outside this standard may change the timestamp and that the SET TIMESTAMP command is illegal.

A timestamp changeable by methods outside this standard (TCMOS) bit set to one specifies that the timestamp may be initialized by methods outside the scope of this standard. A TCMOS bit set to zero specifies that the timestamp shall not be changed by any method except those defined by this standard.

An implicit asymmetric logical unit access enabled (IALUAE) bit set to one specifies that implicitly managed transitions between primary target port asymmetric access states (see 5.9.2) are allowed. An IALUAE bit set to zero specifies that implicitly managed transitions between primary target port asymmetric access states be disallowed and indicates that implicitly managed transitions between primary target port asymmetric access states are disallowed or not supported.

The INITIAL COMMAND PRIORITY field specifies the priority that may be used as the command priority (see SAM-4) for tasks received by the logical unit on any I\_T nexus (i.e., on any I\_T\_L nexus) where a priority has not been modified by a SET PRIORITY command (see 6.34). If a MODE SELECT command specifies an initial command priority value that is different than the current initial command priority, then the device server shall set any priorities that have not be set with a SET PRIORITY command to a value different than the new initial command priority value to the new priority. The device server shall establish a unit attention condition for the initiator port associated with every I\_T\_L nexus that receives a new priority, with the additional sense code set to PRIORITY CHANGED.



### 7.4.8 Disconnect-Reconnect mode page

The Disconnect-Reconnect mode page (see table 351) provides the application client the means to tune the performance of a service delivery subsystem. The name for this mode page, disconnect-reconnect, comes from the SCSI parallel interface. The mode page policy (see 6.9) for this mode page shall be shared or per target port. If the SCSI target device contains more than one target port, the mode page policy should be per target port.

**Table 351 — Disconnect-Reconnect mode page**

Bit Byte	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	PAGE CODE (02h)					
1	PAGE LENGTH (0Eh)							
2	BUFFER FULL RATIO							
3	BUFFER EMPTY RATIO							
4	(MSB)	BUS INACTIVITY LIMIT						
5								(LSB)
6	(MSB)	DISCONNECT TIME LIMIT						
7								(LSB)
8	(MSB)	CONNECT TIME LIMIT						
9								(LSB)
10	(MSB)	MAXIMUM BURST SIZE						
11								(LSB)
12	EMDP	FAIR ARBITRATION			DIMM	DTDC		
13	Reserved							
14	(MSB)	FIRST BURST SIZE						
15								(LSB)

The Disconnect-Reconnect mode page controls parameters that affect one or more target ports. The parameters that may be implemented are specified in the SCSI transport protocol standard (see 3.1.113) for the target port. The MLUS bit (see 7.7.6) shall be set to one in the mode page policy descriptor for this mode page.

The parameters for a target port affect its behavior regardless of which initiator port is forming an I\_T nexus with the target port. The parameters may be accessed by MODE SENSE (see 6.11) and MODE SELECT (see 6.9) commands directed to any logical unit accessible through the target port. If a parameter value is changed, all the device servers for all logical units accessible through the target port shall establish a unit attention condition for the initiator port associated with every I\_T nexus that includes the target port except the I\_T nexus on which the MODE SELECT command was received, with the additional sense code set to MODE PARAMETERS CHANGED.

If a parameter that is not appropriate for the specific SCSI transport protocol implemented by the target port is non-zero, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

An interconnect tenancy is a period of time during which a given pair of SCSI ports (i.e., an initiator port and a target port) are accessing the interconnect layer to communicate with each other (e.g., on arbitrated interconnects, a tenancy typically begins when a SCSI port successfully arbitrates for the interconnect and ends when the SCSI

port releases the interconnect for use by other devices). Data and other information transfers take place during interconnect tenancies.

The PS bit, SPF bit, PAGE CODE field, and PAGE LENGTH field are described in 7.4.5.

The BUFFER FULL RATIO field specifies to the target port how full the buffer should be during read operations prior to requesting an interconnect tenancy. Target ports that do not implement the requested ratio should round down to the nearest implemented ratio as defined in 5.4.

The BUFFER EMPTY RATIO field specifies to the target port how empty the buffer should be during write operations prior to requesting an interconnect tenancy. Target ports that do not implement the requested ratio should round down to the nearest implemented ratio as defined in 5.4.

The buffer full and buffer empty ratios are numerators of a fractional multiplier that has 256 as its denominator. A value of zero indicates that the target port determines when to request an interconnect tenancy consistent with the disconnect time limit parameter. These parameters are advisory to the target port.

NOTE 55 - As an example, consider a target port with ten 512-byte buffers and a specified buffer full ratio of 3Fh. The formula is:  $\text{INTEGER}((\text{ratio} \div 256) \times \text{number of buffers})$ . Therefore in this example  $\text{INTEGER}((3\text{Fh} \div 256) \times 10) = 2$ . During the read operations described in this example, the target port should request an interconnect tenancy whenever two or more buffers are full.

The BUS INACTIVITY LIMIT field specifies the maximum time that the target port is permitted to maintain an interconnect tenancy without data or information transfer. If the bus inactivity limit is exceeded, then the target port shall conclude the interconnect tenancy, within the restrictions placed on it by the applicable SCSI transport protocol. The contents of the DTDC field in this mode page also shall affect the duration of an interconnect tenancy. This value may be rounded as defined in 5.4. A value of zero specifies that there is no bus inactivity limit. Different SCSI transport protocols define different units of measure for the bus inactivity limit.

The DISCONNECT TIME LIMIT field specifies the minimum time that the target port shall wait between interconnect tenancies. This value may be rounded as defined in 5.4. A value of zero specifies that there is no disconnect time limit. Different SCSI transport protocols define different units of measure for the disconnect time limit.

The CONNECT TIME LIMIT field specifies the maximum duration of a single interconnect tenancy. If the connect time limit is exceeded, then the target port shall conclude the interconnect tenancy, within the restrictions placed on it by the applicable SCSI transport protocol. The contents of the DTDC field in this mode page also shall affect the duration of an interconnect tenancy. This value may be rounded as defined in 5.4. A value of zero specifies that there is no connect time limit. Different SCSI transport protocols define different units of measure for the connect time limit.

The MAXIMUM BURST SIZE field indicates the maximum amount of data that the target port shall transfer during a single data transfer operation. This value is expressed in increments of 512 bytes (i.e., a value of one means 512 bytes, two means 1 024 bytes, etc.). The relationship, if any, between data transfer operations and interconnect tenancies is defined in the individual SCSI transport protocol standards. A value of zero specifies there is no limit on the amount of data transferred per data transfer operation.

In terms of the SCSI transport protocol services (see SAM-4), the device server shall limit the Request Byte Count argument to the **Receive Data-Out** protocol service and the **Send Data-In** protocol service to the amount specified in the MAXIMUM BURST SIZE field.

The enable modify data pointers (EMDP) bit specifies whether or not the target port may transfer data out of order. If the EMDP bit is set to zero, the target port shall not transfer data out of order. If the EMDP bit is set to one, the target port is allowed to transfer data out of order.

The FAIR ARBITRATION field specifies whether the target port should use fair or unfair arbitration when requesting an interconnect tenancy. The field may be used to specify different fairness methods as defined in the individual SCSI transport protocol standards.

A disconnect immediate (D IMM) bit set to zero specifies that the target port may transfer data for a command during the same interconnect tenancy in which it receives the command. Whether or not the target port does so may depend upon the target port's internal algorithms, the rules of the applicable SCSI transport protocol, and settings of the other parameters in this mode page. A disconnect immediate (D IMM) bit set to one specifies that the target port shall not transfer data for a command during the same interconnect tenancy in which it receives the command.

The data transfer disconnect control (DTDC) field (see table 352) defines other restrictions on when multiple interconnect tenancies are permitted. A non-zero value in the DTDC field shall take precedence over other interconnect tenancy controls represented by other fields in this mode page.

**Table 352 — Data transfer disconnect control (DTDC) field**

Code	Description
000b	Data transfer disconnect control is not used. Interconnect tenancies are controlled by other fields in this mode page.
001b	All data for a command shall be transferred within a single interconnect tenancy.
010b	Reserved
011b	All data and the response for a command shall be transferred within a single interconnect tenancy.
100b to 111b	Reserved

The FIRST BURST SIZE field specifies the maximum amount of data that may be transferred to the target port for a command along with the command (i.e., the first burst). This value is expressed in increments of 512 bytes (i.e., a value of one means 512 bytes, two means 1 024 bytes, etc.). The meaning of a value of zero is SCSI transport protocol specific. SCSI transport protocols supporting this field shall provide an additional mechanism to enable and disable the first burst function.

In terms of the SCSI transport protocol services (see SAM-4), the **Receive Data-Out** protocol service shall retrieve the first FIRST BURST SIZE amount of data from the first burst.

### 7.4.9 Extended mode page

The Extended mode page (see table 353) provides a means to specify subpages that are defined for all device types. Subpage code 00h is reserved. All Extended mode pages use the sub\_page format.

**Table 353 — Extended mode page**

Bit Byte	7	6	5	4	3	2	1	0
0	PS	SPF (1b)	PAGE CODE (15h)					
1	SUBPAGE CODE							
2	(MSB)	PAGE LENGTH (n-3)						
3	(LSB)							
4	Subpage specific mode parameters							
n								

The PS bit, SPF bit, PAGE CODE field, and PAGE LENGTH field are described in 7.4.5.

### 7.4.10 Extended Device-Type Specific mode page

The Extended Device-Type Specific mode page (see table 354) provides a means to specify subpages that are defined differently for each device type. Subpage code 00h is reserved. All Extended Device-Type Specific mode pages use the sub\_page format.

**Table 354 — Extended Device-Type Specific mode page**

Bit Byte	7	6	5	4	3	2	1	0
0	PS	SPF (1b)	PAGE CODE (16h)					
1	SUBPAGE CODE							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
4	Subpage specific mode parameters							
n								

The PS bit, SPF bit, PAGE CODE field, and PAGE LENGTH field are described in 7.4.5.

### 7.4.11 Informational Exceptions Control mode page

The Informational Exceptions Control mode page (see table 355) defines the methods used by the device server to control the reporting and the operations of specific informational exception conditions. This page shall only apply to informational exceptions that report additional sense codes with the ADDITIONAL SENSE CODE field set to 5Dh (e.g., FAILURE PREDICTION THRESHOLD EXCEEDED) or 0Bh (e.g., WARNING\_ to the application client. The mode page policy (see 6.9) for this mode page shall be shared, or per I\_T nexus.

Informational exception conditions occur as the result of background scan errors, background self-test errors, or vendor specific events within a logical unit. An informational exception condition may occur asynchronous to any commands issued by an application client.

NOTE 56 - Storage devices that support SMART (Self-Monitoring Analysis and Reporting Technology) for predictive failure software should use informational exception conditions.

**Table 355 — Informational Exceptions Control mode page**

Bit Byte	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	PAGE CODE (1Ch)					
1	PAGE LENGTH (0Ah)							
2	PERF	Reserved	EBF	EWASC	DEXCPT	TEST	EBACKERR	LOGERR
3	Reserved				MRIE			
4	(MSB)							
7	INTERVAL TIMER							(LSB)
8	(MSB)							
11	REPORT COUNT							(LSB)

The PS bit, SPF bit, PAGE CODE field, and PAGE LENGTH field are described in 7.4.5.

If the log errors (LOGERR) bit is set to zero, the logging of informational exception conditions by a device server is vendor specific. If the LOGERR bit is set to one, the device server shall log informational exception conditions.

A TEST bit set to one shall create a test device failure at the next interval time, as specified by the INTERVAL TIMER field, if the DEXCPT bit is set to zero. When the TEST bit is set to one, the MRIE and REPORT COUNT fields shall apply as if the TEST bit were zero. The test device failure shall be reported with the additional sense code set to FAILURE PREDICTION THRESHOLD EXCEEDED (FALSE). If both the TEST bit and the DEXCPT bit are set to one, the MODE SELECT command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST. A TEST bit set to zero indicates the device server shall not generate any test device failure notifications.

An enable background error (EBACKERR) bit set to zero indicates the target shall disable reporting of background self-test errors (see 5.6.3.4) and background scan errors (see SBC-3). An EBACKERR bit set to one indicates reporting of background self-test errors and background scan errors shall be enabled. The method for reporting background self-test errors and background scan errors is determined by contents of the MRIE field. Background self-test errors and background scan errors shall be reported as soon as the method specified in the MRIE field occurs (i.e., the INTERVAL TIMER field and REPORT COUNT field do not apply for background self-test errors and background scan errors).

A disable exception control (DEXCPT) bit set to zero indicates the failure prediction threshold exceeded reporting shall be enabled. The method for reporting the failure prediction threshold exceeded when the DEXCPT bit is set to zero is determined from the MRIE field. A DEXCPT bit set to one indicates the device server shall disable reporting of the failure prediction threshold exceeded. The MRIE field is ignored when DEXCPT is set to one and EWASC is set to zero.

If the enable warning (EWASC) bit is set to zero, the device server shall disable reporting of the warning. The MRIE field is ignored when DEXCPT is set to one and the EWASC bit is set to zero. If the EWASC bit is set to one, warning reporting shall be enabled. The method for reporting the warning when the EWASC bit is set to one is determined from the MRIE field.

If background functions are supported and the Enable Background Function (EBF) bit is set to one, then the device server shall enable background functions. If the EBF bit is set to zero, the device server shall disable the functions. Background functions with separate enable control bits (e.g., background medium scan defined in SBC-3) are not controlled by this bit.

For the purposes of the EBF bit, background functions are defined as idle time functions that may impact performance that are performed by a device server operating without errors but do not impact the reliability of the logical unit.

If the performance (PERF) bit is set to zero, informational exception operations that are the cause of delays are acceptable. If the PERF bit is set to one, the device server shall not cause delays while doing informational exception operations. A PERF bit set to one may cause the device server to disable some or all of the informational exceptions operations, thereby limiting the reporting of informational exception conditions.

The value in the method of reporting informational exceptions (MRIE) field (see table 356) defines the method that shall be used by the device server to report informational exception conditions. The priority of reporting multiple information exceptions is vendor specific.

**Table 356 — Method of reporting informational exceptions (MRIE) field (part 1 of 2)**

Code	Description
0h	<b>No reporting of informational exception condition:</b> The device server shall not report information exception conditions.
1h	<b>Asynchronous event reporting:</b> Obsolete
2h	<p><b>Establish unit attention condition:</b> The device server shall report informational exception conditions by establishing a unit attention condition (see SAM-4) for the initiator port associated with every I_T nexus, with the additional sense code set to indicate the cause of the informational exception condition.</p> <p>As defined in SAM-4, the command that has the CHECK CONDITION status with the sense key set to UNIT ATTENTION is not processed before the informational exception condition is reported.</p>
<sup>a</sup> In some command standards (see 3.1.27), this is controlled by the post error (PER) bit in the Read-Write Error Recovery mode page.	

Table 356 — Method of reporting informational exceptions (MRIE) field (part 2 of 2)

Code	Description
3h	<p><b>Conditionally generate recovered error:</b> The device server shall report informational exception conditions, if the reporting of recovered errors is allowed,<sup>a</sup> by returning CHECK CONDITION status. If the TEST bit is set to zero, the status may be returned after the informational exception condition occurs on any command for which GOOD status would have been returned. If the TEST bit is set to one, the status shall be returned on the next command received on any I_T nexus that is normally capable of returning an informational exception condition when the test bit is set to zero. The sense key shall be set to RECOVERED ERROR and the additional sense code shall indicate the cause of the informational exception condition.</p> <p>The command that returns the CHECK CONDITION for the informational exception shall complete without error before any informational exception condition may be reported.</p>
4h	<p><b>Unconditionally generate recovered error:</b> The device server shall report informational exception conditions, regardless of whether the reporting of recovered errors is allowed,<sup>a</sup> by returning CHECK CONDITION status. If the TEST bit is set to zero, the status may be returned after the informational exception condition occurs on any command for which GOOD status would have been returned. If the TEST bit is set to one, the status shall be returned on the next command received on any I_T nexus that is normally capable of returning an informational exception condition when the TEST bit is set to zero. The sense key shall be set to RECOVERED ERROR and the additional sense code shall indicate the cause of the informational exception condition.</p> <p>The command that returns the CHECK CONDITION for the informational exception shall complete without error before any informational exception condition may be reported.</p>
5h	<p><b>Generate no sense:</b> The device server shall report informational exception conditions by returning CHECK CONDITION status. If the TEST bit is set to zero, the status may be returned after the informational exception condition occurs on any command for which GOOD status would have been returned. If the TEST bit is set to one, the status shall be returned on the next command received on any I_T nexus that is normally capable of returning an informational exception condition when the TEST bit is set to zero. The sense key shall be set to NO SENSE and the additional sense code shall indicate the cause of the informational exception condition.</p> <p>The command that returns the CHECK CONDITION for the informational exception shall complete without error before any informational exception condition may be reported.</p>
6h	<p><b>Only report informational exception condition on request:</b> The device server shall preserve the informational exception(s) information. To find out about information exception conditions the application client polls the device server by issuing a REQUEST SENSE command. In the REQUEST SENSE parameter data that contains the sense data, the sense key shall be set to NO SENSE and the additional sense code shall indicate the cause of the informational exception condition.</p>
7h to Bh	Reserved
Ch to Fh	Vendor specific
<sup>a</sup> In some command standards (see 3.1.27), this is controlled by the post error (PER) bit in the Read-Write Error Recovery mode page.	

The INTERVAL TIMER field specifies the period in 100 millisecond increments that the device server shall use for reporting that an informational exception condition has occurred. The device server shall not report informational exception conditions more frequently than the time specified by the INTERVAL TIMER field and shall report them after the time specified by INTERVAL TIMER field has elapsed. After the informational exception condition has been

reported the interval timer shall be restarted. An INTERVAL TIMER field set to zero or FFFF FFFFh specifies that the period for reporting an informational exception condition is vendor specific.

The REPORT COUNT field specifies the maximum number of times the device server may report an informational exception condition to the application client. A REPORT COUNT field set to zero specifies that there is no limit on the number of times the device server may report an informational exception condition.

The maintaining of the interval timer and the report counter across power cycles, hard resets, logical unit resets, and I\_T nexus losses is vendor specific.

#### 7.4.12 Power Condition mode page

The Power Condition mode page provides an application client with methods to control the power condition of a logical unit (see 5.10). These methods include:

- a) Specifying that the logical unit transition to a power condition without delay; and
- b) Activating and setting of idle condition and standby condition timers to specify that the logical unit wait for a period of inactivity before transitioning to a specified power condition.

The mode page policy (see 6.9) for this mode page shall be shared.

When a device server receives a command while in a power condition based on a setting in the Power Condition mode page, the logical unit shall transition to the power condition that allows the command to be processed. If either the idle condition timer or the standby condition timer has been set, then they shall be reset on receipt of the command. On completion of the command, the timer(s) shall be started.

Logical units that contain cache memory shall write all cached data to the medium for the logical unit (e.g., as a logical unit does in response to a SYNCHRONIZE CACHE command as described in SBC-2) prior to entering into any power condition that prevents accessing the media (e.g., before a hard drive stops its spindle motor during transition to the standby power condition).

The logical unit shall use the values in the Power Condition mode page to control its power condition after a power on or a hard reset until a START STOP UNIT command setting a power condition is received.

Table 357 defines the Power Condition mode page.

**Table 357 — Power Condition mode page**

Bit Byte	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	PAGE CODE (1Ah)					
1	PAGE LENGTH (0Ah)							
2	Reserved							
3	Reserved						IDLE	STANDBY
4	(MSB) IDLE CONDITION TIMER (LSB)							
7								
8	(MSB) STANDBY CONDITION TIMER (LSB)							
11								

The PS bit, SPF bit, PAGE CODE field, and PAGE LENGTH field are described in 7.4.5.



The IDLE and STANDBY bits specify which timers are active.

If the IDLE bit is set to one and the STANDBY bit is set to zero, then the idle condition timer is active and the device server shall transition to the idle power condition when the idle condition timer is zero.

If the IDLE bit is set to zero, then the device server shall ignore the idle condition timer.

If the STANDBY bit is set to one and the IDLE bit is set to zero, then the standby condition timer is active and the device server shall transition to the standby power condition when the standby condition timer is zero.

If the STANDBY bit is set to zero, then the device server shall ignore the standby condition timer.

If both the IDLE and STANDBY bits are set to one, then both timers are active and run concurrently. When the idle condition timer is zero the device server shall transition to the idle power condition. When the standby condition timer is zero the device server shall transition to the standby power condition. If the standby condition timer is zero before the idle condition timer is zero, then the logical unit shall transition to the standby power condition.

The value in the IDLE CONDITION TIMER field specifies the inactivity time in 100 millisecond increments that the logical unit shall wait before transitioning to the idle power condition when the IDLE bit is set to one. The idle condition timer is expired when:

- a) The IDLE CONDITION TIMER field is set to zero; or
- b) The number of milliseconds specified by the value in the IDLE CONDITION TIMER field times 100 milliseconds has elapsed since the last activity (e.g., processing a command that requires the active power condition or performing a self test).

The value in the STANDBY CONDITION TIMER field specifies the inactivity time in 100 millisecond increments that the logical unit shall wait before transitioning to the standby power condition when the STANDBY bit is set to one. The standby condition timer is expired when:

- a) The STANDBY CONDITION TIMER field is set to zero; or
- b) The number of milliseconds specified by the value in the STANDBY CONDITION TIMER field times 100 milliseconds has elapsed since the last activity (e.g., processing any command or performing a self test).

#### 7.4.13 Protocol Specific Logical Unit mode page

The Protocol Specific Logical Unit mode page (see table 358) provides protocol specific controls that are associated with a logical unit.

**Table 358 — Protocol Specific Logical Unit mode page**

Bit Byte	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	PAGE CODE (18h)					
1	PAGE LENGTH (n-1)							
2	Reserved				PROTOCOL IDENTIFIER			
3	Protocol specific mode parameters							
n								

During an I\_T\_L nexus, the Protocol Specific Logical Unit mode page controls parameters that affect both:

- a) One or more target ports; and
- b) The logical unit.

The parameters that may be implemented are specified in the SCSI transport protocol standard (see 3.1.113) for the target port. The mode page policy (see 6.9) for this mode page shall be shared or per target port and should be per target port.

The parameters for a target port and logical unit affect their behavior regardless of which initiator port is forming an I\_T\_L nexus with the target port and logical unit. If a parameter value is changed, the device server shall establish a unit attention condition for the initiator port associated with every I\_T nexus except the I\_T nexus on which the MODE SELECT command was received, with the additional sense code set to MODE PARAMETERS CHANGED.

The PS bit, SPF bit, PAGE CODE field, and PAGE LENGTH field are described in 7.4.5.

The value in the PROTOCOL IDENTIFIER field (see 7.5.1) defines the SCSI transport protocol to which the mode page applies. For a MODE SENSE command (see 6.11), the device server shall set the PROTOCOL IDENTIFIER field to one of the values shown in table 361 (see 7.5.1) to indicate the SCSI transport protocol used by the target port through which the MODE SENSE command is being processed. For a MODE SELECT command (see 6.9), the application client shall set the PROTOCOL IDENTIFIER field to one of the values shown in table 361 indicating the SCSI transport protocol to which the protocol specific mode parameters apply. If a device server receives a mode page containing a transport protocol identifier value other than the one used by the target port on which the MODE SELECT command was received, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

7.4.14 Protocol Specific Port mode page

The Protocol Specific Port mode page provides protocol specific controls that are associated with a SCSI port. The page\_0 format (see table 359) is used for subpage 00h and sub\_page format (see table 360) is used for subpages 01h through FEh. See the SCSI transport protocol standard (see 3.1.113) for definition of the protocol specific mode parameters.

Table 359 — Page\_0 format Protocol Specific Port mode page

Bit Byte	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	PAGE CODE (19h)					
1	PAGE LENGTH (n-1)							
2	Reserved				PROTOCOL IDENTIFIER			
3	Protocol specific mode parameters							
n								

**Table 360 — Sub\_page format Protocol Specific Port mode page**

Bit Byte	7	6	5	4	3	2	1	0
0	PS	SPF (1b)	PAGE CODE (19h)					
1	SUBPAGE CODE							
2	(MSB)	PAGE LENGTH (n-3)						(LSB)
3								
4	Reserved							
5	Reserved				PROTOCOL IDENTIFIER			
6	Protocol specific mode parameters							
n								

The Protocol Specific Port mode page controls parameters that affect one or more target ports. The parameters that may be implemented are specified in the SCSI transport protocol standard (see 3.1.113) for the target port. The mode page policy (see 6.9) for this mode page shall be shared or per target port. If the SCSI target device contains more than one target port, the mode page policy should be per target port.

The parameters for a target port affect its behavior regardless of which initiator port is forming an I\_T nexus with the target port. The MLUS bit (see 7.7.6) shall be set to one in the mode page policy descriptor for this mode page.

The parameters may be accessed by MODE SENSE (see 6.11) and MODE SELECT (see 6.9) commands directed to any logical unit accessible through the target port. If a parameter value is changed, the device servers for all logical units accessible through the target port shall establish a unit attention condition for the initiator port associated with every I\_T nexus except the I\_T nexus on which the MODE SELECT command was received, with the additional sense code set to MODE PARAMETERS CHANGED.

The PS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.4.5.

The value in the PROTOCOL IDENTIFIER field (see 7.5.1) defines the SCSI transport protocol to which the mode page applies. For a MODE SENSE command, the device server shall set the PROTOCOL IDENTIFIER field to one of the values shown in table 361 (see 7.5.1) to indicate the SCSI transport protocol used by the target port through which the MODE SENSE command is being processed. For a MODE SELECT command, the application client shall set the PROTOCOL IDENTIFIER field to one of the values shown in table 361 indicating the SCSI transport protocol to which the protocol specific mode parameters apply. If a device server receives a mode page containing a transport protocol identifier value other than the one used by the target port on which the MODE SELECT command was received, then command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

## 7.5 Protocol specific parameters

### 7.5.1 Protocol specific parameters introduction

Some commands use protocol specific information in their CDBs or parameter lists. This subclause describes those protocol specific parameters.

Protocol specific parameters may include a PROTOCOL IDENTIFIER field (see table 361) as a reference for the SCSI transport protocol to which the protocol specific parameter applies.

**Table 361 — PROTOCOL IDENTIFIER values**

<b>Protocol Identifier</b>	<b>Description</b>	<b>Protocol Standard</b>
0h	Fibre Channel	FCP-2
1h	Parallel SCSI	SPI-5
2h	SSA	SSA-S3P
3h	IEEE 1394	SBP-3
4h	SCSI Remote Direct Memory Access Protocol	SRP
5h	Internet SCSI (iSCSI)	iSCSI
6h	SAS Serial SCSI Protocol	SAS
7h	Automation/Drive Interface Transport Protocol	ADT
8h	AT Attachment Interface (ATA/ATAPI)	ATA/ATAPI-7
9h to Eh	Reserved	
Fh	No specific protocol	

### 7.5.2 Alias entry protocol specific designations

#### 7.5.2.1 Introduction to alias entry protocol specific designations

The alias entries (see 6.2.2) in the parameter data for the CHANGE ALIASES command (see 6.2) and REPORT ALIASES command (see 6.21) include FORMAT CODE, DESIGNATION LENGTH and DESIGNATION fields whose contents and meaning are based on the SCSI transport protocol specified in a PROTOCOL IDENTIFIER field (see 7.5.1). This subclause defines the SCSI transport protocol specific format codes, designation lengths, and designations.

### 7.5.2.2 Fibre Channel specific alias entry designations

#### 7.5.2.2.1 Introduction to Fibre Channel specific alias entry designations

If an alias entry **PROTOCOL IDENTIFIER** field contains the Fibre Channel protocol identifier (0h, see table 361), the **FORMAT CODE**, **DESIGNATION LENGTH** and **DESIGNATION** fields shall be as defined in table 362.

**Table 362 — Fibre Channel alias entry format codes**

Format Code	Description	Designation Length (bytes)	Designation Subclause
00h	World Wide Port Name	8	7.5.2.2.2
01h	World Wide Port Name with N_Port checking	12	7.5.2.2.3
02h to FFh	Reserved		

#### 7.5.2.2.2 Fibre Channel world wide port name alias entry designation

If the **PROTOCOL IDENTIFIER** and **FORMAT CODE** fields specify a Fibre Channel world wide port name designation, the alias entry **DESIGNATION** field shall have the format shown in table 363.

**Table 363 — Fibre Channel world wide port name alias entry designation**

Bit Byte	7	6	5	4	3	2	1	0
0	See table 96 in 6.2.2.							
15								
16	FIBRE CHANNEL WORLD WIDE PORT NAME							
23								

The **FIBRE CHANNEL WORLD WIDE PORT NAME** field shall contain the port world wide name defined by the port login (PLOGI) extended link service (see FC-FS).

A Fibre Channel world wide port name designation is valid (see 6.2.3) if the device server has access to a SCSI domain formed by a Fibre Channel fabric and the fabric contains a port with the specified port world wide name.

### 7.5.2.2.3 Fibre Channel world wide port name with N\_Port checking alias entry designation

If the PROTOCOL IDENTIFIER and FORMAT CODE fields specify a Fibre Channel world wide port name with N\_Port checking designation, the alias entry DESIGNATION field shall have the format shown in table 364.

**Table 364 — Fibre Channel world wide port name with N\_Port checking alias entry designation**

Bit Byte	7	6	5	4	3	2	1	0
0	See table 96 in 6.2.2.							
15								
16	FIBRE CHANNEL WORLD WIDE PORT NAME							
23								
24	Reserved							
25	(MSB)	N_PORT						(LSB)
27								

The FIBRE CHANNEL WORLD WIDE PORT NAME field shall contain the port world wide name defined by the port login (PLOGI) extended link service (see FC-FS).

The N\_PORT field shall contain the FC\_FS port D\_ID to be used to transport frames including PLOGI and FCP-2 related frames.

A Fibre Channel world wide port name with N\_Port checking designation is valid (see 6.2.3) if all of the following conditions are true:

- The device server has access to a SCSI domain formed by a Fibre Channel fabric;
- The fabric contains a port with the specified port World Wide Name; and
- The value in the N\_PORT field is the N\_Port identifier of a Fibre Channel port whose port world wide name matches that in the FIBRE CHANNEL WORLD WIDE PORT NAME field.

### 7.5.2.3 RDMA specific alias entry designations

#### 7.5.2.3.1 Introduction to RDMA specific alias entry designations

If an alias entry PROTOCOL IDENTIFIER field contains the SCSI RDMA protocol identifier (4h, see table 361), the FORMAT CODE, DESIGNATION LENGTH and DESIGNATION fields shall be as defined in table 365.

**Table 365 — RDMA alias entry format codes**

Format Code	Description	Designation Length (bytes)	Designation Subclause
00h	Target Port Identifier	16	7.5.2.3.2
01h	InfiniBand™ Global Identifier with Target Port Identifier checking	32	7.5.2.3.3
02h to FFh	Reserved		

### 7.5.2.3.2 RDMA target port identifier alias entry designation

If the PROTOCOL IDENTIFIER and FORMAT CODE fields specify a SCSI RDMA target port identifier designation, the alias entry DESIGNATION field shall have the format shown in table 366.

**Table 366 — RDMA target port identifier alias entry designation**

Bit Byte	7	6	5	4	3	2	1	0
0	See table 96 in 6.2.2.							
15								
16	TARGET PORT IDENTIFIER							
31								

The TARGET PORT IDENTIFIER field shall contain an SRP target port identifier.

A SCSI RDMA target port identifier designation is valid (see 6.2.3) if the device server has access to an SRP SCSI domain containing the specified SRP target port identifier.

### 7.5.2.3.3 InfiniBand global identifier with target port identifier checking alias entry designation

If the PROTOCOL IDENTIFIER and FORMAT CODE fields specify an InfiniBand global identifier with target port identifier checking designation, the alias entry designation field shall have the format shown in table 367.

**Table 367 — InfiniBand global identifier with target port identifier checking alias entry designation**

Bit Byte	7	6	5	4	3	2	1	0
0	See table 96 in 6.2.2.							
15								
16	INFINIBAND GLOBAL IDENTIFIER							
31								
32	TARGET PORT IDENTIFIER							
47								

The INFINIBAND GLOBAL IDENTIFIER field contains an InfiniBand global identifier (GID) of an InfiniBand port connected to an SRP target port.

The TARGET PORT IDENTIFIER field shall contain an SRP target port identifier.

An InfiniBand global identifier with target port identifier checking designation is valid (see 6.2.3) if all of the following conditions are true:

- The device server has access to an SRP SCSI domain layered on InfiniBand;
- The device server has access to an SRP target port based on the InfiniBand global identifier specified in the INFINIBAND GLOBAL IDENTIFIER field; and
- The value in the TARGET PORT IDENTIFIER field is the SRP target port identifier for the SRP target port that is accessible via the InfiniBand global identifier contained in the INFINIBAND GLOBAL IDENTIFIER field.

### 7.5.2.4 Internet SCSI specific alias entry designations

#### 7.5.2.4.1 Introduction to Internet SCSI specific alias entry designations

If an alias entry **PROTOCOL IDENTIFIER** field contains the iSCSI protocol identifier (5h, see table 361), the **FORMAT CODE**, **DESIGNATION LENGTH** and **DESIGNATION** fields shall be as defined in table 368.

**Table 368 — iSCSI alias entry format codes**

<b>Format Code</b>	<b>Description</b>	<b>Designation Length (bytes, maximum)</b>	<b>Designation Subclause</b>
00h	iSCSI Name	224	7.5.2.4.2
01h	iSCSI Name with binary IPv4 address	236	7.5.2.4.3
02h	iSCSI Name with IPName	488	7.5.2.4.4
03h	iSCSI Name with binary IPv6 address	248	7.5.2.4.5
04h to FFh	Reserved		

NOTE 57 - A designation that contains no IP addressing information or contains IP addressing information that does not address the named SCSI target device may require a device server to have access to a name server or to other discovery protocols to resolve the given iSCSI Name to an IP address through which the device server may establish iSCSI Login. Access to such a service is protocol specific and vendor specific.

#### 7.5.2.4.2 iSCSI name alias entry designation

If the **PROTOCOL IDENTIFIER** and **FORMAT CODE** fields specify iSCSI name designation, the alias entry **DESIGNATION** field shall have the format shown in table 369.

**Table 369 — iSCSI name alias entry designation**

<b>Bit Byte</b>	7	6	5	4	3	2	1	0
0	See table 96 in 6.2.2.							
15								
16	(MSB)	iSCSI NAME						
4m-1								(LSB)

The null-terminated, null-padded (see 4.4.2) iSCSI NAME field shall contain the iSCSI name of an iSCSI node (see RFC 3720). The number of bytes in the iSCSI NAME field shall be a multiple of four.

An iSCSI name designation is valid if the device server has access to a SCSI domain containing an Internet protocol network and that network contains an iSCSI node with the specified iSCSI name.



#### 7.5.2.4.3 iSCSI name with binary IPv4 address alias entry designation

If the PROTOCOL IDENTIFIER and FORMAT CODE fields specify iSCSI name with binary IPv4 address designation, the alias entry DESIGNATION field shall have the format shown in table 370.

**Table 370 — iSCSI name with binary IPv4 address alias entry designation**

Bit Byte	7	6	5	4	3	2	1	0
0	See table 96 in 6.2.2.							
15								
16	(MSB)	ISCSI NAME						(LSB)
4m-1								
4m	(MSB)	IPV4 ADDRESS						(LSB)
4m+3								
4m+4	Reserved							
4m+5	Reserved							
4m+6	(MSB)	PORT NUMBER						(LSB)
4m+7								
4m+8	Reserved							
4m+9	Reserved							
4m+10	(MSB)	INTERNET PROTOCOL NUMBER						(LSB)
4m+11								

The null-terminated, null-padded (see 4.4.2) ISCSI NAME field shall contain the iSCSI name of an iSCSI node (see RFC 3720). The number of bytes in the ISCSI NAME field shall be a multiple of four.

The IPV4 ADDRESS field shall contain an IPv4 address (see RFC 791).

The PORT NUMBER field shall contain a TCP port number (see 3.1.174). The TCP port number shall conform to the requirements defined by iSCSI (see RFC 3720).

The INTERNET PROTOCOL NUMBER field shall contain an Internet protocol number (see 3.1.75). The Internet protocol number shall conform to the requirements defined by iSCSI (see RFC 3720).

An iSCSI name designation is valid if the device server has access to a SCSI domain containing an Internet protocol network and that network contains an iSCSI node with the specified iSCSI name.

The IPv4 address, port number, and Internet protocol number provided in the designation may be used by a device server for addressing to discover and establish communication with the named iSCSI node. Alternatively, the device server may use other protocol specific or vendor specific methods to discover and establish communication with the named iSCSI node.

#### 7.5.2.4.4 iSCSI name with IPname alias entry designation

If the PROTOCOL IDENTIFIER and FORMAT CODE fields specify iSCSI name with IPname designation, the alias entry DESIGNATION field shall have the format shown in table 371.

**Table 371 — iSCSI name with IPname alias entry designation**

Bit Byte	7	6	5	4	3	2	1	0
0	See table 96 in 6.2.2.							
15								
16	(MSB)	ISCSI NAME						(LSB)
k								
k+1	(MSB)	IPNAME						(LSB)
n								
n+1	PAD (if needed)							
4m-1								
4m	Reserved							
4m+1	Reserved							
4m+2	(MSB)	PORT NUMBER						(LSB)
4m+3								
4m+4	Reserved							
4m+5	Reserved							
4m+6	(MSB)	INTERNET PROTOCOL NUMBER						(LSB)
4m+7								

The null-terminated (see 4.4.2) ISCSI NAME field shall contain the iSCSI name of an iSCSI node (see RFC 3720).

The null-terminated (see 4.4.2) IPNAME field shall contain a Internet protocol domain name (see 3.1.74).

The PAD field shall contain zero to three bytes set to zero such that the total length of the ISCSI NAME, IPNAME, and PAD fields is a multiple of four. Device servers shall ignore the PAD field.

The PORT NUMBER field shall contain a TCP port number (see 3.1.174). The TCP port number shall conform to the requirements defined by iSCSI (see RFC 3720).

The INTERNET PROTOCOL NUMBER field shall contain an Internet protocol number (see 3.1.75). The Internet protocol number shall conform to the requirements defined by iSCSI (see RFC 3720).

An iSCSI name designation is valid if the device server has access to a SCSI domain containing an Internet protocol network and that network contains an iSCSI node with the specified iSCSI name.

The Internet protocol domain name, port number, and Internet protocol number provided in the designation may be used by a device server for addressing to discover and establish communication with the named iSCSI node. Alternatively, the device server may use other protocol specific or vendor specific methods to discover and establish communication with the named iSCSI node.

#### 7.5.2.4.5 iSCSI name with binary IPv6 address alias entry designation

If the PROTOCOL IDENTIFIER and FORMAT CODE fields specify iSCSI name with binary IPv6 address designation, the alias entry DESIGNATION field shall have the format shown in table 372.

**Table 372 — iSCSI name with binary IPv6 address alias entry designation**

Bit Byte	7	6	5	4	3	2	1	0
0	See table 96 in 6.2.2.							
15								
16	(MSB)	ISCSI NAME						(LSB)
n								
4m	(MSB)	IPV6 ADDRESS						(LSB)
4m+15								
4m+16	Reserved							
4m+17	Reserved							
4m+18	(MSB)	PORT NUMBER						(LSB)
4m+19								
4m+20	Reserved							
4m+21	Reserved							
4m+22	(MSB)	INTERNET PROTOCOL NUMBER						(LSB)
4m+23								

The null-terminated, null-padded (see 4.4.2) ISCSI NAME field shall contain the iSCSI name of an iSCSI node (see RFC 3720).

The IPV6 ADDRESS field shall contain an IPv6 address (see RFC 2373).

The PORT NUMBER field shall contain a TCP port number (see 3.1.174). The TCP port number shall conform to the requirements defined by iSCSI (see RFC 3720).

The INTERNET PROTOCOL NUMBER field shall contain an Internet protocol number (see 3.1.75). The Internet protocol number shall conform to the requirements defined by iSCSI (see RFC 3720).

An iSCSI name designation is valid if the device server has access to a SCSI domain containing an Internet protocol network and that network contains an iSCSI node with the specified iSCSI name.

The IPv6 address, port number and Internet protocol number provided in the designation may be used by a device server for addressing to discover and establish communication with the named iSCSI node. Alternatively, the device server may use other protocol specific or vendor specific methods to discover and establish communication with the named iSCSI node.

### 7.5.3 EXTENDED COPY protocol specific target descriptors

#### 7.5.3.1 Introduction to EXTENDED COPY protocol specific target descriptors

The protocol-specific target descriptors in the parameter data of the EXTENDED COPY command (see 6.3) are described in this subclause. An introduction to EXTENDED COPY target descriptors is provided in 6.3.6.1.

NOTE 58 - In the EXTENDED COPY command the target in target descriptor refers to a copy target device (i.e., the source or destination of an EXTENDED COPY operation), not a SCSI target device. Target descriptors specify logical unit numbers or proxy tokens and may also specify initiator ports, target ports, and SCSI target devices used to access those logical units.

#### 7.5.3.2 Fibre Channel N\_Port\_Name EXTENDED COPY target descriptor format

The target descriptor format shown in table 373 is used by an EXTENDED COPY command to specify an FCP copy target device using its Fibre Channel N\_Port\_Name.

**Table 373 — Fibre Channel N\_Port\_Name EXTENDED COPY target descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E0h)							
1	LU ID TYPE		NUL	PERIPHERAL DEVICE TYPE				
2	(MSB)							
3	RELATIVE INITIATOR PORT IDENTIFIER							(LSB)
4	LU IDENTIFIER							
11								
12	N_PORT_NAME							
19								
20	Reserved							
27								
28	Device type specific parameters							
31								

The DESCRIPTOR TYPE CODE field, LU ID TYPE field, PERIPHERAL DEVICE TYPE field, NUL bit, RELATIVE INITIATOR PORT IDENTIFIER field, LU IDENTIFIER field, and the device type specific parameters are described in 6.3.6.1.

The N\_PORT\_NAME field shall contain the N\_Port\_Name defined by the port login (PLOGI) extended link service (see FC-FS).

NOTE 59 - The N\_Port\_Name EXTENDED COPY target descriptor format necessitates translating the N\_Port\_Name to an N\_Port\_ID (see FC-FS and 7.5.3.3).

### 7.5.3.3 Fibre Channel N\_Port\_ID EXTENDED COPY target descriptor format

The target descriptor format shown in table 374 is used by an EXTENDED COPY command to specify an FCP copy target device using its Fibre Channel N\_Port\_ID.

**Table 374 — Fibre Channel N\_Port\_ID EXTENDED COPY target descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E1h)							
1	LU ID TYPE		NUL	PERIPHERAL DEVICE TYPE				
2	(MSB)							
3	RELATIVE INITIATOR PORT IDENTIFIER							(LSB)
4								
11	LU IDENTIFIER							
12								
20	Reserved							
21	(MSB)							
22	N_PORT_ID							
23								(LSB)
24								
27	Reserved							
28								
31	Device type specific parameters							

The DESCRIPTOR TYPE CODE field, LU ID TYPE field, PERIPHERAL DEVICE TYPE field, NUL bit, RELATIVE INITIATOR PORT IDENTIFIER field, LU IDENTIFIER field, and the device type specific parameters are described in 6.3.6.1.

The N\_PORT\_ID field shall contain the port D\_ID (see FC-FS) to be used to transport frames including PLOGI and FCP-2 related frames.

NOTE 60 - Use of N\_Port\_ID addressing restricts this target descriptor format to a single Fibre Channel fabric.

### 7.5.3.4 Fibre Channel N\_Port\_ID with N\_Port\_Name checking EXTENDED COPY target descriptor format

The target descriptor format shown in table 375 is used by an EXTENDED COPY command to specify an FCP copy target device using its Fibre Channel N\_Port\_ID and to require the copy manager to verify that the N\_Port\_Name of the specified N\_Port matches the value in the target descriptor.

**Table 375 — Fibre Channel N\_Port\_ID with N\_Port\_Name checking target descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E2h)							
1	LU ID TYPE		NUL	PERIPHERAL DEVICE TYPE				
2	(MSB)							
3	RELATIVE INITIATOR PORT IDENTIFIER (LSB)							
4								
11	LU IDENTIFIER							
12								
19	N_PORT_NAME							
20	Reserved							
21	(MSB)							
22								
23	N_PORT_ID (LSB)							
24								
27	Reserved							
28								
31	Device type specific parameters							

The DESCRIPTOR TYPE CODE field, LU ID TYPE field, PERIPHERAL DEVICE TYPE field, NUL bit, RELATIVE INITIATOR PORT IDENTIFIER field, LU IDENTIFIER field, and the device type specific parameters are described in 6.3.6.1.

The N\_PORT\_NAME field shall contain the N\_Port\_Name defined by the port login (PLOGI) extended link service (see FC-FS).

The N\_PORT field shall contain the port D\_ID (see FC-FS) to be used to transport frames including PLOGI and FCP-2 related frames.

NOTE 61 - Use of N\_Port addressing restricts this target descriptor format to a single fabric.

When the copy manager first processes a segment descriptor that references this target descriptor, it shall confirm that the D\_ID in the N\_PORT\_ID field is associated with the N\_Port\_Name in the N\_PORT\_NAME field. If the confirmation fails, the command shall be terminated because the copy target device is unavailable (see 6.3.3). The SCSI device processing this target descriptor shall track configuration changes that affect the D\_ID value for the duration of the task. An application client is responsible for tracking configuration changes between commands.

### 7.5.3.5 SCSI Parallel T\_L EXTENDED COPY target descriptor format

The target descriptor format shown in table 376 is used by an EXTENDED COPY command to specify a SPI copy target device using its SCSI target identifier.

**Table 376 — SCSI Parallel T\_L EXTENDED COPY target descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E3h)							
1	LU ID TYPE		NUL	PERIPHERAL DEVICE TYPE				
2	(MSB)							
3	RELATIVE INITIATOR PORT IDENTIFIER							(LSB)
4								
11	LU IDENTIFIER							
12	Vendor specific							
13	TARGET IDENTIFIER							
14								
27	Reserved							
28								
31	Device type specific parameters							

The DESCRIPTOR TYPE CODE field, LU ID TYPE field, PERIPHERAL DEVICE TYPE field, NUL bit, RELATIVE INITIATOR PORT IDENTIFIER field, LU IDENTIFIER field, and the device type specific parameters are described in 6.3.6.1.

The TARGET IDENTIFIER field specifies the SCSI target identifier (see SPI-5).

### 7.5.3.6 IEEE 1394 EUI-64 EXTENDED COPY target descriptor format

The target descriptor format shown in table 377 is used by an EXTENDED COPY command to specify an SBP copy target device using its IEEE 1394 Extended Unique Identifier, 64-bits (EUI-64) and configuration ROM (Read-Only Memory) directory identifier.

**Table 377 — IEEE 1394 EUI-64 EXTENDED COPY target descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E8h)							
1	LU ID TYPE		NUL	PERIPHERAL DEVICE TYPE				
2	(MSB)							
3	RELATIVE INITIATOR PORT IDENTIFIER							(LSB)
4								
11	LU IDENTIFIER							
12								
19	EUI-64							
20								
22	DIRECTORY ID							
23								
27	Reserved							
28								
31	Device type specific parameters							

The DESCRIPTOR TYPE CODE field, LU ID TYPE field, PERIPHERAL DEVICE TYPE field, NUL bit, RELATIVE INITIATOR PORT IDENTIFIER field, LU IDENTIFIER field, and the device type specific parameters are described in 6.3.6.1.

The EUI-64 field shall contain the node's unique identifier (EUI-64) obtained from the configuration ROM bus information block, as specified by ANSI IEEE 1394a:2000.

NOTE 62 - ANSI IEEE 1394a-2000 separately labels the components of the EUI-64 as NODE\_VENDOR\_ID, CHIP\_ID\_HI and CHIP\_ID\_LO. Collectively these form the node's EUI-64.

The DIRECTORY ID field shall contain the copy target device's directory identifier, as specified by ISO/IEC 13213:1994.



### 7.5.3.7 RDMA EXTENDED COPY target descriptor format

The target descriptor format shown in table 378 is used by an EXTENDED COPY command to specify an SRP copy target device using its RDMA SRP target port identifier.

**Table 378 — RDMA EXTENDED COPY target descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E7h)							
1	LU ID TYPE		NUL	PERIPHERAL DEVICE TYPE				
2	(MSB)							
3	RELATIVE INITIATOR PORT IDENTIFIER							(LSB)
4								
11	LU IDENTIFIER							
12								
27	TARGET PORT IDENTIFIER							
28								
31	Device type specific parameters							

The DESCRIPTOR TYPE CODE field, LU ID TYPE field, PERIPHERAL DEVICE TYPE field, NUL bit, RELATIVE INITIATOR PORT IDENTIFIER field, LU IDENTIFIER field, and the device type specific parameters are described in 6.3.6.1.

The TARGET PORT IDENTIFIER field contains the SRP target port identifier (see SRP).

### 7.5.3.8 iSCSI binary IPv4 address EXTENDED COPY target descriptor format

The target descriptor format shown in table 379 is used by an EXTENDED COPY command to specify an iSCSI copy target device using its binary IPv4 (Internet Protocol version 4) address.

**Table 379 — iSCSI binary IPv4 address EXTENDED COPY target descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E5h)							
1	LU ID TYPE		NUL	PERIPHERAL DEVICE TYPE				
2	(MSB) _____							
3	RELATIVE INITIATOR PORT IDENTIFIER _____ (LSB)							
4	_____							
11	LU IDENTIFIER _____							
12	(MSB) _____							
15	IPV4 ADDRESS _____ (LSB)							
16	_____							
21	Reserved _____							
22	(MSB) _____							
23	PORT NUMBER _____ (LSB)							
24	Reserved _____							
25	Reserved _____							
26	(MSB) _____							
27	INTERNET PROTOCOL NUMBER _____ (LSB)							
28	_____							
31	Device type specific parameters _____							

The DESCRIPTOR TYPE CODE field, LU ID TYPE field, PERIPHERAL DEVICE TYPE field, NUL bit, RELATIVE INITIATOR PORT IDENTIFIER field, LU IDENTIFIER field, and the device type specific parameters are described in 6.3.6.1.

The IPV4 ADDRESS field shall contain an IPv4 address (see RFC 791).

The PORT NUMBER field shall contain the TCP port number (see 3.1.174). The TCP port number shall conform to the requirements defined by iSCSI (see RFC 3720).

The INTERNET PROTOCOL NUMBER field shall contain an Internet protocol number (see 3.1.75). The Internet protocol number shall conform to the requirements defined by iSCSI (see RFC 3720).

### 7.5.3.9 SAS serial SCSI protocol target descriptor format

The target descriptor format shown in table 380 is used by an EXTENDED COPY command to specify a SAS copy target device using its SAS address.

**Table 380 — SAS serial SCSI protocol EXTENDED COPY target descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E9h)							
1	LU ID TYPE		NUL	PERIPHERAL DEVICE TYPE				
2	(MSB) _____							
3	RELATIVE INITIATOR PORT IDENTIFIER _____ (LSB)							
4	_____							
11	LU IDENTIFIER _____							
12	_____							
19	SAS ADDRESS _____							
20	_____							
27	Reserved _____							
28	_____							
31	Device type specific parameters _____							

The DESCRIPTOR TYPE CODE field, LU ID TYPE field, PERIPHERAL DEVICE TYPE field, NUL bit, RELATIVE INITIATOR PORT IDENTIFIER field, LU IDENTIFIER field, and the device type specific parameters are described in 6.3.6.1.

The SAS ADDRESS field contains the SAS address (see SAS).

### 7.5.4 TransportID identifiers

#### 7.5.4.1 Overview of TransportID identifiers

An application client may use a TransportID to specify an initiator port other than the initiator port that is transporting the command and parameter data (e.g., as an Access identifiers (see 8.3.1.3.2) in ACL ACEs, as the initiator port in the I\_T nexus to which PERSISTENT RESERVE OUT command with REGISTER AND MOVE service action (see 5.7.8) is moving a persistent reservation).

TransportIDs (see table 381) shall be at least 24 bytes long and shall be a multiple of four bytes in length.

**Table 381 — TransportID format**

Bit Byte	7	6	5	4	3	2	1	0
0	FORMAT CODE		Reserved		PROTOCOL IDENTIFIER			
1								
n	SCSI transport protocol specific data							

The FORMAT CODE field specifies the format of the TransportID. All format codes not specified in this standard are reserved.

The PROTOCOL IDENTIFIER field (see table 361 in 7.5.1) specifies the SCSI transport protocol to which the TransportID applies.

The format of the SCSI transport protocol specific data depends on the value in the PROTOCOL IDENTIFIER field. The SCSI transport protocol specific data in a TransportID shall only include initiator port identifiers, initiator port names, or SCSI device names (see SAM-4) that persist across hard resets and I\_T nexus losses. TransportID formats specific to SCSI transport protocols are listed in table 382.

**Table 382 — TransportID formats for specific SCSI transport protocols**

SCSI transport Protocol	Protocol Standard	Reference
Fibre Channel	FCP-2	7.5.4.2
Parallel SCSI	SPI-5	7.5.4.3
IEEE 1394	SBP-3	7.5.4.4
Remote Direct Memory Access (RDMA)	SRP	7.5.4.5
Internet SCSI (iSCSI)	iSCSI	7.5.4.6
SAS Serial SCSI Protocol	SAS	7.5.4.7

#### 7.5.4.2 TransportID for initiator ports using SCSI over Fibre Channel

A Fibre Channel TransportID (see table 383) specifies an FCP-2 initiator port based on the N\_Port\_Name belonging to that initiator port.

**Table 383 — Fibre Channel TransportID format**

Bit Byte	7	6	5	4	3	2	1	0
0	FORMAT CODE (00b)		Reserved		PROTOCOL IDENTIFIER (0h)			
1	Reserved							
7								
8	N_PORT_NAME							
15								
16	Reserved							
23								

The N\_PORT\_NAME field shall contain the N\_Port\_Name defined by the Physical Log In (PLOGI) extended link service (see FC-FS).

### 7.5.4.3 TransportID for initiator ports using a parallel SCSI bus

A parallel SCSI bus TransportID (see table 384) specifies a SPI-5 initiator port based on the SCSI address of an initiator port and the relative port identifier of the target port through which the application client accesses the SCSI target device.

**Table 384 — Parallel SCSI bus TransportID format**

Bit Byte	7	6	5	4	3	2	1	0
0	FORMAT CODE (00b)		Reserved		PROTOCOL IDENTIFIER (1h)			
1	Reserved							
2	(MSB)							
3	SCSI ADDRESS (LSB)							
4	Obsolete							
5								
6	(MSB)							
7	RELATIVE TARGET PORT IDENTIFIER (LSB)							
8								
23	Reserved							

The SCSI ADDRESS field specifies the SCSI address (see SPI-5) of the initiator port.

The RELATIVE TARGET PORT IDENTIFIER field specifies the relative port identifier (see 3.1.120) of the target port for which the initiator port SCSI address applies. If the RELATIVE TARGET PORT IDENTIFIER does not reference a target port in the SCSI target device, the TransportID is invalid.

### 7.5.4.4 TransportID for initiator ports using SCSI over IEEE 1394

An IEEE 1394 TransportID (see table 385) specifies an SBP-3 initiator port based on the EUI-64 initiator port name belonging to that initiator port.

**Table 385 — IEEE 1394 TransportID format**

Bit Byte	7	6	5	4	3	2	1	0
0	FORMAT CODE (00b)		Reserved		PROTOCOL IDENTIFIER (3h)			
1	Reserved							
7								
8	EUI-64 NAME							
15								
16	Reserved							
23								

The EUI-64 NAME field shall contain the EUI-64 IEEE 1394 node unique identifier (see SBP-3) for an initiator port.

#### 7.5.4.5 TransportID for initiator ports using SCSI over an RDMA interface

A RDMA TransportID (see table 386) specifies an SRP initiator port based on the world wide unique initiator port name belonging to that initiator port.

**Table 386 — RDMA TransportID format**

Bit Byte	7	6	5	4	3	2	1	0
0	FORMAT CODE (00b)		Reserved		PROTOCOL IDENTIFIER (4h)			
1	Reserved							
7								
8	INITIATOR PORT IDENTIFIER							
23								

The INITIATOR PORT IDENTIFIER field shall contain an SRP initiator port identifier (see SRP).

#### 7.5.4.6 TransportID for initiator ports using SCSI over iSCSI

An iSCSI TransportID specifies an iSCSI initiator port using one of the TransportID formats listed in table 387.

**Table 387 — iSCSI TransportID formats**

Format code	Description
00b	Initiator port is identified using the world wide unique SCSI device name of the iSCSI initiator device containing the initiator port (see table 388).
01b	Initiator port is identified using the world wide unique initiator port identifier (see table 389).
10b to 11b	Reserved

iSCSI TransportIDs with a format code of 00b may be rejected. iSCSI TransportIDs with a format code of 01b should not be rejected.

A iSCSI TransportID with the format code set to 00b (see table 388) specifies an iSCSI initiator port based on the world wide unique SCSI device name of the iSCSI initiator device containing the initiator port.

**Table 388 — iSCSI initiator device TransportID format**

Bit Byte	7	6	5	4	3	2	1	0
0	FORMAT CODE (00b)		Reserved		PROTOCOL IDENTIFIER (5h)			
1	Reserved							
2	(MSB) _____							
3	ADDITIONAL LENGTH (m-3) _____							
4	(MSB) _____							
m	ISCSI NAME _____							
	(LSB)							

The ADDITIONAL LENGTH field specifies the number of bytes that follow in the TransportID. The additional length shall be at least 20 and shall be a multiple of four.

The null-terminated, null-padded (see 4.4.2) iSCSI NAME field shall contain the iSCSI name of an iSCSI initiator node (see RFC 3720). The first iSCSI NAME field byte containing an ASCII null character terminates the iSCSI NAME field without regard for the specified length of the iSCSI TransportID or the contents of the ADDITIONAL LENGTH field.

NOTE 63 - The maximum length of the iSCSI TransportID is 228 bytes because the iSCSI name length does not exceed 223 bytes.

If a iSCSI TransportID with the format code set to 00b appears in a PERSISTENT RESERVE OUT parameter list (see 6.14.3), all initiator ports known to the device server with an iSCSI node name matching the one in the TransportID shall be registered.

If a iSCSI TransportID with the format code set to 00b appears in an ACE access identifier (see 8.3.1.3.2), the logical units listed in the ACE shall be accessible to any initiator port with an iSCSI node name matching the value in the TransportID. The access controls coordinator shall reject any command that attempts to define more than one ACEs with an iSCSI TransportID access identifier containing the same iSCSI name. The command shall be terminated with CHECK CONDITION status, with the sense key ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

A iSCSI TransportID with the format code set to 01b (see table 389) specifies an iSCSI initiator port based on its world wide unique initiator port identifier.

**Table 389 — iSCSI initiator port TransportID format**

Bit Byte	7	6	5	4	3	2	1	0
0	FORMAT CODE (01b)		Reserved		PROTOCOL IDENTIFIER (5h)			
1	Reserved							
2	(MSB)							
3	ADDITIONAL LENGTH (m-3)							(LSB)
4	(MSB)							
n-1	ISCSI NAME							(LSB)
n	(MSB)							
n+4	SEPARATOR (2C 692C 3078h)							(LSB)
n+5	(MSB)							
m	ISCSI INITIATOR SESSION ID							(LSB)

The ADDITIONAL LENGTH field specifies the number of bytes that follow in the TransportID encompassing the iSCSI NAME, SEPARATOR, and iSCSI INITIATOR SESSION ID fields. The additional length shall be at least 20 and shall be a multiple of four.

The iSCSI NAME field shall contain the iSCSI name of an iSCSI initiator node (see RFC 3720). The iSCSI NAME field shall not be null-terminated (see 4.4.2) and shall not be padded.

The SEPARATOR field shall contain the five ASCII characters ",i,0x".

NOTE 64 - The notation used to define the SEPARATOR field is described in 3.6.1.

The null-terminated, null-padded iSCSI INITIATOR SESSION ID field shall contain the iSCSI initiator session identifier (see RFC 3720) in the form of ASCII characters that are the hexadecimal digits converted from the binary iSCSI initiator session identifier value. The first iSCSI INITIATOR SESSION ID field byte containing an ASCII null character

terminates the iSCSI INITIATOR SESSION ID field without regard for the specified length of the iSCSI TransportID or the contents of the ADDITIONAL LENGTH field.

#### 7.5.4.7 TransportID for initiator ports using SCSI over SAS Serial SCSI Protocol

A SAS Serial SCSI Protocol (SSP) TransportID (see table 390) specifies a SAS initiator port that is communicating via SSP using the SAS address belonging to that initiator port.

**Table 390 — SAS Serial SCSI Protocol TransportID format**

Bit Byte	7	6	5	4	3	2	1	0
0	FORMAT CODE (00b)		Reserved		PROTOCOL IDENTIFIER (6h)			
1	Reserved							
3								
4	SAS ADDRESS							
11								
12	Reserved							
23								

The SAS ADDRESS field specifies the SAS address of the initiator port.



## 7.6 Security protocol parameters

### 7.6.1 Security protocol information description

#### 7.6.1.1 Overview

The purpose of security protocol information security protocol (i.e., the SECURITY PROTOCOL field set to 00h in a SECURITY PROTOCOL IN command) is to transfer security protocol related information from the logical unit. A SECURITY PROTOCOL IN command in which the SECURITY PROTOCOL field is set to 00h is not associated with an previous SECURITY PROTOCOL OUT command and shall be processed without regard for whether a SECURITY PROTOCOL OUT command has been processed.

If the SECURITY PROTOCOL IN command is supported, the SECURITY PROTOCOL value of 00h shall be supported as defined in this standard.

#### 7.6.1.2 CDB description

When the SECURITY PROTOCOL field is set to 00h in a SECURITY PROTOCOL IN command, the SECURITY PROTOCOL SPECIFIC field (see table 391) contains a single numeric value as defined in 3.5.

**Table 391 — SECURITY PROTOCOL SPECIFIC field for SECURITY PROTOCOL IN protocol 00h**

Code	Description	Support	Reference
0000h	Supported security protocol list	Mandatory	7.6.1.3
0001h	Certificate data	Mandatory	7.6.1.4
0002h to FFFFh	Reserved		

All other CDB fields for SECURITY PROTOCOL IN command shall meet the requirements stated in 6.30.

Each time a SECURITY PROTOCOL IN command with the SECURITY PROTOCOL field set to 00h is received, the device server shall transfer the data defined 7.6.1 starting with byte 0.

### 7.6.1.3 Supported security protocols list description

If the SECURITY PROTOCOL field is set to 00h and the SECURITY PROTOCOL SPECIFIC field is set to 0000h in a SECURITY PROTOCOL IN command, the parameter data shall have the format shown in table 392.

**Table 392 — Supported security protocols SECURITY PROTOCOL IN parameter data**

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved							
5								
6	(MSB)	SUPPORTED SECURITY PROTOCOL LIST LENGTH (m-7)						(LSB)
7								
Supported security protocol list								
8	SUPPORTED SECURITY PROTOCOL [first] (00h)							
	⋮							
m	SUPPORTED SECURITY PROTOCOL [last]							
m+1	Pad bytes (optional)							
n								

The SUPPORTED SECURITY PROTOCOL LIST LENGTH field indicates the total length, in bytes, of the supported security protocol list that follows.

Each SUPPORTED SECURITY PROTOCOL field in the supported security protocols list shall contain one of the security protocol values supported by the logical unit. The values shall be listed in ascending order starting with 00h.

The total data length shall conform to the ALLOCATION LENGTH field requirements (see 6.30). Pad bytes may be appended to meet this length. Pad bytes shall have a value of 00h.

### 7.6.1.4 Certificate data description

#### 7.6.1.4.1 Certificate overview

A certificate is either an X.509 Public Key Certificate (see 7.6.1.4.2) or an X.509 Attribute Certificate (see 7.6.1.4.3) depending on the capabilities of the logical unit.

If the SECURITY PROTOCOL field is set to 00h and the SECURITY PROTOCOL SPECIFIC field is set to 0001h in a SECURITY PROTOCOL IN command, the parameter data shall have the format shown in table 393.

**Table 393 — Certificate data SECURITY PROTOCOL IN parameter data**

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved							
1								
2	(MSB)	CERTIFICATE LENGTH (m-3)						
3								(LSB)
4	CERTIFICATE							
m								
m+1	Pad bytes (optional)							
n								

The CERTIFICATE LENGTH field indicates the total length, in bytes, of the certificate or certificates that follow. The length may include more than one certificates. If the device server doesn't have a certificate to transfer, the CERTIFICATE LENGTH field shall be set to 0000h.

The contents of the CERTIFICATE field are defined in 7.6.1.4.2 and 7.6.1.4.3.

The total data length shall conform to the ALLOCATION LENGTH field requirements (see 6.30). Pad bytes may be appended to meet this length. Pad bytes shall have a value of 00h.

#### 7.6.1.4.2 Public Key certificate description

RFC 3280 defines the certificate syntax for certificates consistent with X.509v3 Public Key Certificate Specification. Any further restrictions beyond the requirements of RFC 3280 are yet to be defined by T10.

#### 7.6.1.4.3 Attribute certificate description

RFC 3281 defines the certificate syntax for certificates consistent with X.509v2 Attribute Certificate Specification. Any further restrictions beyond the requirements of RFC 3281 are yet to be defined by T10.

## 7.6.2 SA creation capabilities

### 7.6.2.1 Overview

If the SECURITY PROTOCOL field in a SECURITY PROTOCOL IN command (see 6.30) is set to 40h, then the command returns information related to the SA creation (see 5.14.2.3) capabilities provided by the device server.

The SA creation capabilities protocol is independent of any other SA creation protocols. The device server shall not refuse to process an SA creation capabilities SECURITY PROTOCOL IN command, and this processing shall not affect the state maintained for any SA creation CCS (e.g., an IKEv2-SCSI CCS) on any I\_T\_L nexus. Except for those cases where an SA creation capabilities SECURITY PROTOCOL IN command reports changed SA creation capabilities, processing of the command shall not affect the concurrent processing of any commands that are part of an SA creation CCS.

If any SA creation protocols are supported, the SA creation capabilities protocol shall be supported as described in 7.6.2.

The SA creation capabilities SECURITY PROTOCOL IN CDB format is described in 7.6.2.2.

As shown in table 394 (see 7.6.2.2), the format of the parameter data returned by a SA creation capabilities SECURITY PROTOCOL IN command depends on the value in the SECURITY PROTOCOL SPECIFIC field in the CDB.

### 7.6.2.2 SA creation capabilities CDB description

The SA creation capabilities SECURITY PROTOCOL IN CDB has the format defined in 6.30 with the additional requirements described in this subclause.

When the SECURITY PROTOCOL field is set to SA creation capabilities (i.e., 40h) in a SECURITY PROTOCOL IN command, the SECURITY PROTOCOL SPECIFIC field (see table 394) identifies the SA creation protocol (see 5.14.2.3) for which the device server shall return capability information.

**Table 394 — SECURITY PROTOCOL SPECIFIC field for the SA creation capabilities SECURITY PROTOCOL IN command**

Code	Description	Parameter data format
0000h	Supported device server capabilities formats	7.6.2.3.1
0001h to 0100h	Reserved	
0101h	IKEv2-SCSI device server capabilities	
0102h to EFFFh	Reserved	
F000h to FFFFh	Vendor Specific	

If an SA creation capabilities SECURITY PROTOCOL IN command is received with the INC\_512 bit is set to one, then the SECURITY PROTOCOL IN command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

### 7.6.2.3 SA creation capabilities parameter data formats

#### 7.6.2.3.1 Supported device server capabilities formats parameter data format

The supported device server capabilities formats parameter data (see table 395) indicates which device server capabilities parameter data formats are supported by the device server.

**Table 395 — Supported device server capabilities formats parameter data**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
3	PARAMETER DATA LENGTH (n-3) (LSB)							
Supported capabilities parameter data formats								
4	CAPABILITIES PARAMETER DATA FORMAT [first] (0000h)							
5								
	⋮							
n-1	CAPABILITIES PARAMETER DATA FORMAT [last]							
n								

The PARAMETER DATA LENGTH field indicates the number of bytes that follow in the parameter data.

Each CAPABILITIES PARAMETER DATA FORMAT field in the supported capabilities parameter data formats list shall contain one of the SECURITY PROTOCOL SPECIFIC field values (see table 394) supported by the device server. The values shall be listed in ascending order starting with 0000h.

#### 7.6.2.3.2 IKEv2-SCSI device server capabilities parameter data format

The IKEv2-SCSI device server capabilities parameter data (see table 396) indicates the IKEv2 transforms (i.e., key exchange protocols and authentication protocols) supported by the device server for IKEv2-SCSI.

**Table 396 — IKEv2-SCSI device server capabilities parameter data**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____							
3	PARAMETER DATA LENGTH (n-3) _____ (LSB)							
4	_____							
n	IKEv2-SCSI SA Creation Capabilities payload (see 7.6.3.5.11) _____							

The PARAMETER DATA LENGTH field indicates the number of bytes that follow in the parameter data.

The IKEv2-SCSI SA Creation Capabilities payload (see 7.6.3.5.11) indicates the algorithms supported by IKEv2-SCSI in the Key Exchange step (see 5.14.4.8) and Authentication step (see 5.14.4.9).

NOTE 65 - The primary content of the IKEv2-SCSI device server capabilities SA creation capabilities parameter data is an IKEv2-SCSI payload (see 7.6.3.5) because the IKEv2-SCSI SA Creation Capabilities payload is used by the device server and application client in the construction of other IKEv2-SCSI payloads (see 5.14.4).

### 7.6.3 IKEv2-SCSI

#### 7.6.3.1 Overview

If the SECURITY PROTOCOL field in a SECURITY PROTOCOL OUT command (see 6.31) or a SECURITY PROTOCOL IN command (see 6.30) is set to 41h, then the command is part of an IKEv2-SCSI CCS (see 5.14.4) and is used to transfer IKEv2-SCSI protocol information to or from the device server.

In an IKEv2-SCSI CCS, a defined sequence of SECURITY PROTOCOL OUT and SECURITY PROTOCOL IN commands are sent by the application client and processed by the device server as summarized in 5.14.4.1.

The IKEv2-SCSI SECURITY PROTOCOL OUT CDB format is described in 7.6.3.3.

The IKEv2-SCSI SECURITY PROTOCOL IN CDB format is described in 7.6.3.2.

The IKEv2-SCSI SECURITY PROTOCOL OUT command and the IKEv2-SCSI SECURITY PROTOCOL IN command use the same parameter data format, and this format is described in 7.6.3.4. A significant component of the IKEv2-SCSI parameter data format is one or more IKE payloads, and the format of IKE payloads is described in 7.6.3.5.

If the IKEv2-SCSI SA creation protocol is supported (see 7.6.1), the SA creation capabilities protocol (see 7.6.2) shall also be supported.

#### 7.6.3.2 IKEv2-SCSI SECURITY PROTOCOL IN CDB description

The IKEv2-SCSI SECURITY PROTOCOL IN CDB has the format defined in 6.30 with the additional requirements described in this subclause.

When the SECURITY PROTOCOL field is set to IKEv2-SCSI (i.e., 41h) in a SECURITY PROTOCOL IN command, the SECURITY PROTOCOL SPECIFIC field (see table 397) identifies the IKEv2-SCSI step (see 5.14.4.1) that the device server is to process. If the IKEv2-SCSI SA creation protocol is supported (see 7.6.1), the SECURITY PROTOCOL IN command support requirements are shown in table 397.

**Table 397 — SECURITY PROTOCOL SPECIFIC field for the IKEv2-SCSI SECURITY PROTOCOL IN command**

Code	Description	Support	References	
			Usage	Data format
0000h to 00FFh	Restricted		RFC 4306	RFC 4306
0100h to 0101h	Reserved			
0102h	Key Exchange step	Mandatory	5.14.4.8.3	7.6.3.4
0103h	Authentication step	Mandatory	5.14.4.9.3	7.6.3.4
0104h to EFFFh	Reserved			
F000h to FFFFh	Vendor Specific			

If an IKEv2-SCSI SECURITY PROTOCOL IN command is received with the INC\_512 bit is set to one while the device server is maintaining state for an IKEv2-SCSI CCS on the I\_T\_L nexus on which the command was received (see 5.14.4.1), then:

- a) The SECURITY PROTOCOL IN command shall be terminated with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS; and

- b) The device server shall continue the IKEv2-SCSI CCS by preparing to receive another SECURITY PROTOCOL IN command or SECURITY PROTOCOL OUT command, as appropriate.

If an IKEv2-SCSI SECURITY PROTOCOL IN command is received with the INC\_512 bit set to one while the device server is not maintaining state for an IKEv2-SCSI CCS (see 5.14.4.1), then the SECURITY PROTOCOL IN command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

### 7.6.3.3 IKEv2-SCSI SECURITY PROTOCOL OUT CDB description

The IKEv2-SCSI SECURITY PROTOCOL OUT CDB has the format defined in 6.31 with the additional requirements described in this subclause.

When the SECURITY PROTOCOL field is set to IKEv2-SCSI (i.e., 41h) in a SECURITY PROTOCOL OUT command, the SECURITY PROTOCOL SPECIFIC field (see table 398) identifies the IKEv2-SCSI step (see 5.14.4.1) that the device server is to process. If the IKEv2-SCSI SA creation protocol is supported (see 7.6.1), the SECURITY PROTOCOL IN command support requirements are shown in table 398.

**Table 398 — SECURITY PROTOCOL SPECIFIC field for the IKEv2-SCSI SECURITY PROTOCOL OUT command**

Code	Description	Support	References	
			Usage	Data format
0000h to 00FFh	Restricted		RFC 4306	RFC 4306
0100h to 0101h	Reserved			
0102h	Key Exchange step	Mandatory	5.14.4.8.3	7.6.3.4
0103h	Authentication step	Mandatory	5.14.4.9.3	7.6.3.4
0104h	Delete operation	Mandatory	5.14.4.13	7.6.3.4
0105h to EFFFh	Reserved			
F000h to FFFFh	Vendor Specific			

If an IKEv2-SCSI SECURITY PROTOCOL OUT command is received with the INC\_512 bit is set to one while the device server is maintaining state for an IKEv2-SCSI CCS on the I\_T\_L nexus on which the command was received (see 5.14.4.1), then:

- The SECURITY PROTOCOL OUT command shall be terminated with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS; and
- The device server shall continue the IKEv2-SCSI CCS by preparing to receive another SECURITY PROTOCOL OUT command or SECURITY PROTOCOL IN command, as appropriate.

If an IKEv2-SCSI SECURITY PROTOCOL OUT command is received with the INC\_512 bit is set to one while the device server is not maintaining state for an IKEv2-SCSI CCS (see 5.14.4.1), then the SECURITY PROTOCOL OUT command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

Any IKEv2-SCSI SECURITY PROTOCOL OUT command with an transfer length of up to 16 384 bytes shall not be terminated with an error due to the number of bytes to be transferred and processed.

### 7.6.3.4 IKEv2-SCSI parameter data format

Table 399 shows the parameter list format used by a SECURITY PROTOCOL OUT command and the parameter data format used by a SECURITY PROTOCOL IN command when the SECURITY PROTOCOL field is set to IKEv2-SCSI (i.e., 41h).

**Table 399 — IKEv2-SCSI SECURITY PROTOCOL OUT and SECURITY PROTOCOL IN parameter data**

Bit Byte	7	6	5	4	3	2	1	0
IKEv2-SCSI header								
0	Restricted (see RFC 4306)							
3								
4	(MSB)	IKE_SA APPLICATION CLIENT SAI						(LSB)
7								
8	Restricted (see RFC 4306)							
11								
12	(MSB)	IKE_SA DEVICE SERVER SAI						(LSB)
15								
16	NEXT PAYLOAD							
17	MAJOR VERSION (2h)				MINOR VERSION			
18	EXCHANGE TYPE							
19	Reserved			INTTR	VERSION	RSPNS	Reserved	
20	(MSB)	MESSAGE ID						(LSB)
23								
24	(MSB)	IKE LENGTH (n+1)						(LSB)
27								
IKEv2-SCSI payloads								
28	IKEv2-SCSI payload [first] (see 7.6.3.5)							
	⋮							
n	IKEv2-SCSI payload [last] (see 7.6.3.5)							

The IKE\_SA APPLICATION CLIENT SAI field contains the value that is or is destined to become the AC\_SAI SA parameter (see 5.14.2.2) in the generated SA (see 5.14.4.11). The AC\_SAI is chosen by the application client to uniquely identify its representation of the SA that is being negotiated or managed (e.g., deleted).

If the device server receives an IKEv2-SCSI header with the IKE\_SA APPLICATION CLIENT SAI field set to zero, then the error shall be processed as described in 7.6.3.8.

To increase procedural integrity checking, the application client should compare the IKE\_SA APPLICATION CLIENT SAI field contents in any SECURITY PROTOCOL IN parameter data it receives to the value that the application client is



maintaining for the IKEv2-SCSI CCS or SA management. If the two values are not identical, the application client should abandon the IKEv2-SCSI CCS, if any, and notify the device server that the IKEv2-SCSI CCS, if any, is being abandoned as described in 5.14.4.12.

Except in the Key Exchange step SECURITY PROTOCOL OUT command (see 5.14.4.8.2), the IKE\_SA DEVICE SERVER SAI field contains the value that is or is destined to become the DS\_SAI SA parameter in the generated SA. The DS\_SAI is chosen by the device server in accordance with the requirements in 5.14.2.1 to uniquely identify its representation of the SA that is being negotiated. In the Key Exchange step SECURITY PROTOCOL OUT command the IKE\_SA DEVICE SERVER SAI field is reserved.

To increase procedural integrity checking, the application client should compare the IKE\_SA DEVICE SERVER SAI field contents in the Authentication step SECURITY PROTOCOL IN parameter data it receives to the value that the application client is maintaining for the IKEv2-SCSI CCS. If the two values are not identical, the application client should abandon the IKEv2-SCSI CCS and notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 5.14.4.12.

The device server shall validate the contents of the IKE\_SA APPLICATION CLIENT SAI field and the IKE\_SA DEVICE SERVER SAI field as shown in table 400.

**Table 400 — IKEv2-SCSI header checking of SAIs**

Contents of SECURITY PROTOCOL SPECIFIC field in SECURITY PROTOCOL OUT CDB	Expected contents for ...		Device server action if expected field contents not received
	IKE_SA APPLICATION CLIENT SAI field	IKE_SA DEVICE SERVER SAI field	
0102h (i.e., Key Exchange step)	any value	reserved	No actions taken based on expected field contents
0103h (i.e., Authentication step)	A match with the SAI values maintained for an IKEv2-SCSI CCS on the I_T_L nexus on which the command was received		a) The SECURITY PROTOCOL OUT command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to SA CREATION PARAMETER VALUE REJECTED; and b) The device server shall continue the IKEv2-SCSI CCS by preparing to receive another Authentication step SECURITY PROTOCOL OUT command
0104h (i.e., Delete operation)	If at least one IKEv2-SCSI CCS is being maintained for the I_T_L nexus on which the command was received, then: a) A match with the SAI values maintained for an IKEv2-SCSI CCS; or b) A match with the SAI values maintained for any active SA		a) The SECURITY PROTOCOL OUT command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to SA CREATION PARAMETER VALUE REJECTED; and b) The device server shall continue the IKEv2-SCSI CCS by preparing to receive another Authentication step SECURITY PROTOCOL OUT command
	If no IKEv2-SCSI CCS is being maintained for the I_T_L nexus on which the command was received, then a match with the SAI values maintained for any active SA		The SECURITY PROTOCOL OUT command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST

The NEXT PAYLOAD field (see table 401) identifies the first IKEv2-SCSI payload that follows the IKEv2-SCSI header.

**Table 401 — NEXT PAYLOAD field**

Code	IKE Payload Name	Support requirements in SECURITY PROTOCOL ...		Reference
		IN	OUT	
00h	No Next Payload	Mandatory		7.6.3.5.2
01h to 20h		Reserved		
21h	Security Association	Prohibited <sup>a</sup>		RFC 4306
22h	Key Exchange	Mandatory		7.6.3.5.3
23h	Identification – Application Client	Prohibited	Mandatory	7.6.3.5.4
24h	Identification – Device Server	Mandatory	Prohibited	7.6.3.5.4
25h	Certificate	Optional		7.6.3.5.5
26h	Certificate Request	Optional		7.6.3.5.5
27h	Authentication	Mandatory		7.6.3.5.6
28h	Nonce	Mandatory		7.6.3.5.7
29h	Notify	Prohibited	Mandatory	7.6.3.5.8
2Ah	Delete	Prohibited	Mandatory	7.6.3.5.9
2Bh	Vendor ID	Prohibited		RFC 4306
2Ch	Traffic Selector – Application Client	Prohibited		RFC 4306
2Dh	Traffic Selector – Device Server	Prohibited		RFC 4306
2Eh	Encrypted	Mandatory		7.6.3.5.10
2Fh	Configuration	Prohibited		RFC 4306
30h	Extensible Authentication	Prohibited		RFC 4306
31h to 7Fh		Restricted		RFC 4306
80h	IKEv2-SCSI SA Creation Capabilities	Mandatory		7.6.3.5.11
81h	IKEv2-SCSI SA Cryptographic Algorithms	Mandatory		7.6.3.5.12
82h	IKEv2-SCSI SAUT Cryptographic Algorithms	Mandatory		7.6.3.5.13
83h	IKEv2-SCSI Timeout Values	Mandatory		7.6.3.5.14
84h to BFh		Reserved		
C0h to FFh	Vendor Specific			
<sup>a</sup> The Security Association payload type value is not used in IKEv2-SCSI. The IKEv2-SCSI SA Cryptographic Algorithms payload (i.e., 81h) and IKEv2-SCSI SAUT Cryptographic Algorithms payload (i.e., 82h) are used instead.				

The MAJOR VERSION field shall contain the value 2h. If a device server receives an IKEv2-SCSI header with a MAJOR VERSION field containing a value other than 2h, then the error shall be processed as described in 7.6.3.8.

The MINOR VERSION field is reserved.

The EXCHANGE TYPE field is reserved.

The initiator (INTTR) bit shall be set to:

- a) One for SECURITY PROTOCOL OUT commands; and
- b) Zero for SECURITY PROTOCOL IN commands.

If a device server receives an IKEv2-SCSI header with the INTTR bit set to zero, then the error shall be processed as described in 7.6.3.8.

If an application client receives an IKEv2-SCSI header with the INTTR bit set to one, it should abandon the IKEv2-SCSI CCS and notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 5.14.4.12.

The VERSION bit is reserved.

The response (RSPNS) bit shall be set to:

- a) Zero for SECURITY PROTOCOL OUT commands; and
- b) One for SECURITY PROTOCOL IN commands.

If a device server receives an IKEv2-SCSI header with the RSPNS bit set to one, then the error shall be processed as described in 7.6.3.8.

If an application client receives an IKEv2-SCSI header with the RSPNS bit set to zero, it should abandon the IKEv2-SCSI CCS and notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 5.14.4.12.

The MESSAGE ID field (see table 402) identifies the function of the parameter data.

**Table 402 — MESSAGE ID field**

Code	Description
0000 0000h	Key Exchange step SECURITY PROTOCOL OUT command (see 5.14.4.8.2) or SECURITY PROTOCOL IN command (see 5.14.4.8.3)
0000 0001h	Authentication step SECURITY PROTOCOL OUT command (see 5.14.4.9.2) or SECURITY PROTOCOL IN command (see 5.14.4.9.3)
0000 0002h	Delete operation in a SECURITY PROTOCOL OUT command
0000 0003h to 0000 FFFFh	Reserved

If the device server receives a SECURITY PROTOCOL OUT command with an invalid MESSAGE ID field in its IKEv2-SCSI header, then the error shall be processed as described in 7.6.3.8.

If the application client receives an invalid MESSAGE ID field in the parameter data for a SECURITY PROTOCOL IN command, the application client should abandon the IKEv2-SCSI CCS and notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 5.14.4.12.

The IKE LENGTH field contains the total number of bytes in the parameter data, including the IKEv2-SCSI header and all the IKEv2-SCSI payloads.

NOTE 66 - The contents of the IKE LENGTH field differ from those found in most SCSI length fields, however, they are consistent with the IKEv2 usage (see RFC 4306).

Each IKEv2-SCSI payload (see 7.6.3.5) contains specific data related to the operation being performed. A specific combination of IKEv2-SCSI payloads is specified for each operation (e.g., Key Exchange) as summarized in 5.14.4.2. The Encryption payload (see 7.6.3.6.2) nests one set of IKEv2-SCSI payloads inside another.

Based on the contents of the SECURITY PROTOCOL SPECIFIC field in the SECURITY PROTOCOL OUT CDB, the device server shall:

- a) Validate the contents of the NEXT PAYLOAD field in the IKEv2-SCSI header as shown in table 403, before performing any decryption or integrity checking; and
- b) Validate that the number of instances of each next payload value shown in table 403 occur in the parameter data, after the encrypted data, if any, is decrypted and integrity checked (i.e., if the NEXT PAYLOAD field in the IKEv2-SCSI header contains 2Eh, after the Encrypted payload is decrypted and integrity checked).

If the next payload values in the IKEv2-SCSI header and the unencrypted SECURITY PROTOCOL OUT parameter data do not meet the requirements shown in table 403 for the listed SECURITY PROTOCOL SPECIFIC field contents, then the error shall be processed as described in 7.6.3.8.

Based on the contents of the SECURITY PROTOCOL SPECIFIC field in the SECURITY PROTOCOL IN CDB, the device server shall:

- a) Place one of the next payload values allowed by table 403 in the NEXT PAYLOAD field in the IKEv2-SCSI header; and
- b) Include the number of instances of each next payload value shown in table 403 in the parameter data either before or after encryption, if applicable (i.e., if the NEXT PAYLOAD field in the IKEv2-SCSI header contains 2Eh, before the contents of the Encrypted payload are encrypted and integrity check value is computed).

**Table 403 — Next payload values in SECURITY PROTOCOL OUT/IN parameter data (part 1 of 3)**

Next payload value	Next payload value allowed in IKEv2-SCSI header	Number of times a next payload value is allowed
<b>SECURITY PROTOCOL OUT command with SECURITY PROTOCOL SPECIFIC field set to 0102h (i.e., Key Exchange step)</b>		<b>Authentication step not skipped (see 5.14.4.1)</b>
83h (i.e., IKEv2-SCSI Timeout Values)	Yes	1
81h (i.e., IKEv2-SCSI SA Cryptographic Algorithms)	Yes	1
82h (i.e., IKEv2-SCSI SAUT Cryptographic Algorithms)	No	0
22h (i.e., Key Exchange)	Yes	1
28h (i.e., Nonce)	Yes	1
00h (i.e., No Next Payload)	No	1
All other next payload codes	No	0

**Table 403 — Next payload values in SECURITY PROTOCOL OUT/IN parameter data (part 2 of 3)**

Next payload value	Next payload value allowed in IKEv2-SCSI header	Number of times a next payload value is allowed
<b>SECURITY PROTOCOL OUT command with SECURITY PROTOCOL SPECIFIC field set to 0102h (i.e., Key Exchange step)</b>		<b>Authentication step skipped (see 5.14.4.1)</b>
83h (i.e., IKEv2-SCSI Timeout Values)	Yes	1
81h (i.e., IKEv2-SCSI SA Cryptographic Algorithms)	Yes	1
82h (i.e., IKEv2-SCSI SAUT Cryptographic Algorithms)	Yes	1
22h (i.e., Key Exchange)	Yes	1
28h (i.e., Nonce)	Yes	1
00h (i.e., No Next Payload)	No	1
All other next payload codes	No	0
<b>SECURITY PROTOCOL IN command with SECURITY PROTOCOL SPECIFIC field set to 0102h (i.e., Key Exchange step)</b>		<b>Authentication step not skipped (see 5.14.4.1)</b>
81h (i.e., IKEv2-SCSI SA Cryptographic Algorithms)	Yes	1
82h (i.e., IKEv2-SCSI SAUT Cryptographic Algorithms)	No	0
22h (i.e., Key Exchange)	Yes	1
28h (i.e., Nonce)	Yes	1
26h (i.e., Certificate Request)	Yes	0 or more
00h (i.e., No Next Payload)	No	1
All other next payload codes	No	0
<b>SECURITY PROTOCOL IN command with SECURITY PROTOCOL SPECIFIC field set to 0102h (i.e., Key Exchange step)</b>		<b>Authentication step skipped (see 5.14.4.1)</b>
81h (i.e., IKEv2-SCSI SA Cryptographic Algorithms)	Yes	1
82h (i.e., IKEv2-SCSI SAUT Cryptographic Algorithms)	Yes	1
22h (i.e., Key Exchange)	Yes	1
28h (i.e., Nonce)	Yes	1
26h (i.e., Certificate Request)	Yes	0 or more
00h (i.e., No Next Payload)	No	1
All other next payload codes	No	0

**Table 403 — Next payload values in SECURITY PROTOCOL OUT/IN parameter data (part 3 of 3)**

Next payload value	Next payload value allowed in IKEv2-SCSI header	Number of times a next payload value is allowed
<b>SECURITY PROTOCOL OUT command with SECURITY PROTOCOL SPECIFIC field set to 0103h (i.e., Authentication step)</b>		
2Eh (i.e., Encrypted)	Yes	1
23h (i.e., Identification – Application Client)	No	1
82h (i.e., IKEv2-SCSI SAUT Cryptographic Algorithms)	No	1
25h (i.e., Certificate)	No	0 or more
26h (i.e., Certificate Request)	No	0 or more
29h (i.e., Notify)	No	0 or 1
27h (i.e., Authentication)	No	1
00h (i.e., No Next Payload)	No	1
All other next payload codes	No	0
<b>SECURITY PROTOCOL IN command with SECURITY PROTOCOL SPECIFIC field set to 0103h (i.e., Authentication step)</b>		
2Eh (i.e., Encrypted)	Yes	1
24h (i.e., Identification – Device Server)	No	1
82h (i.e., IKEv2-SCSI SAUT Cryptographic Algorithms)	No	1
25h (i.e., Certificate)	No	0 or more
27h (i.e., Authentication)	No	1
00h (i.e., No Next Payload)	No	1
All other next payload codes	No	0
<b>SECURITY PROTOCOL OUT command with SECURITY PROTOCOL SPECIFIC field set to 0104h (i.e., Delete operation)</b>		
2Eh (i.e., Encrypted)	Yes	1
2Ah (i.e., Delete)	No	1
00h (i.e., No Next Payload)	No	1
All other next payload codes	No	0

7.6.3.5 IKEv2-SCSI payloads

7.6.3.5.1 IKEv2-SCSI payload format

Each IKEv2-SCSI payload (see table 404) is composed of a header and data that is specific to the payload type.

Table 404 — IKEv2-SCSI payload format

Bit Byte	7	6	5	4	3	2	1	0
IKEv2-SCSI payload header								
0	NEXT PAYLOAD							
1	CRIT	Reserved						
2	IKE PAYLOAD LENGTH (n+1)							
3								
IKEv2-SCSI payload-specific data								
4								
n								

The NEXT PAYLOAD field identifies the IKEv2-SCSI payload that follows this IKEv2-SCSI payload using one of the code values shown in table 401 (see 7.6.3.4).

If a device server receives an IKEv2-SCSI payload that it does not recognize (e.g., an IKEv2-SCSI payload identified by a next payload value of 01h) with the critical (CRIT) bit set to one, then the error shall be processed as described in 7.6.3.8.

If an application client receives an IKEv2-SCSI payload that it does not recognize with the CRIT bit set to one, then the application client should abandon the IKEv2-SCSI CCS and notify the device server that it is abandoning the IKEv2-SCSI CCS as described in 5.14.4.12.

If an application client or device server receives an IKEv2-SCSI payload that it does not recognize with the CRIT bit set to zero, then it should use the NEXT PAYLOAD field and the IKE PAYLOAD LENGTH field to skip processing of the unrecognized IKEv2-SCSI payload and continue processing at the next IKEv2-SCSI payload.

The IKE PAYLOAD LENGTH field contains the total number of bytes in the payload, including the IKEv2-SCSI payload header. The value in the IKE PAYLOAD LENGTH field need not be a multiple of two or four (i.e., no byte alignment is maintained among IKEv2-SCSI payloads).

NOTE 67 - The contents of the IKE PAYLOAD LENGTH field differ from those found in most SCSI length fields, however, they are consistent with the IKEv2 usage (see RFC 4306).

The format and contents of the IKEv2-SCSI payload-specific data depends on the value in the NEXT PAYLOAD field of:

- a) The IKEv2-SCSI header (see 7.6.3.4), if this is the first IKEv2-SCSI payload in the parameter data; or
- b) The previous IKEv2-SCSI payload, in all other cases.



### 7.6.3.5.2 No Next payload

A NEXT PAYLOAD field that is set to 00h (i.e., No Next payload) specifies that no more IKEv2-SCSI payloads follow the current payload. The IKEv2-SCSI No Next payload contains no bytes and has no format.

### 7.6.3.5.3 Key Exchange payload

The Key Exchange payload (see table 405) transfers Diffie-Hellman shared key exchange data between an application client and a device server or vice versa.

**Table 405 — Key Exchange payload format**

Bit Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT (1b)	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								
4	(MSB)	DIFFIE-HELLMAN GROUP NUMBER						
5								
6		Reserved						
7								
8		KEY EXCHANGE DATA						
n								

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.6.3.5.1.

The CRIT bit is set to one in the Key Exchange payload.

The DIFFIE-HELLMAN GROUP NUMBER field contains the least significant 16 bits from ALGORITHM IDENTIFIER field in the D-H IKEv2-SCSI algorithm descriptor (see 7.6.3.6.5) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.6.3.5.12).

The KEY EXCHANGE DATA field contains the sender's Diffie-Hellman public value for this key exchange. The format of key exchange data is as specified in the reference cited in that registry for the value used.

When a prime modulus (i.e., mod  $p$ ) Diffie-Hellman group is used, the length of the Diffie-Hellman public value shall be equal to the length of the prime modulus over which the exponentiation was performed as shown in table 429 (see 7.6.3.6.5). Zero bits shall be prepended to the KEY EXCHANGE DATA field if necessary.

Diffie-Hellman exponential reuse and reuse of analogous Diffie-Hellman public values for Diffie-Hellman mechanisms not based on exponentiation should not be performed, but if performed it shall be constrained (e.g., requirements regarding when Diffie-Hellman information is discarded) as specified in RFC 4306. The freshness and randomness of the random nonces are critical to the security of IKEv2-SCSI when Diffie-Hellman exponentials and public values are reused (see RFC 4306).

7.6.3.5.4 Identification – Application Client payload and Identification – Device Server payload

The Identification – Application Client payload (see table 406) transfers identification information from the application client to the device server. The Identification – Device Server payload (see table 406) transfers identification information from the device server to the application client.

Table 406 — Identification payload format

Bit Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT (1b)	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4	ID TYPE							
5	Reserved							
7								
8	IDENTIFICATION DATA							
n								

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.6.3.5.1.

The CRIT bit is set to one in the Identification – Application Client payload and the Identification – Device Server payload.

The ID TYPE field describes the contents of the IDENTIFICATION DATA field and shall contain one of the named values shown in table 407.

Table 407 — ID TYPE field

Code	Name <sup>a</sup>	Contents of the IDENTIFICATION DATA field
09h	ID_DER_ASN1_DN	The value of a certificate subject field (see RFC 3280)
0Ah	ID_DER_ASN1_GN	The value of a name contained in a Subject Alternative Name (i.e., SubjectAltName) certificate extension (see RFC 3280)
0Bh	ID_KEY_ID	Arbitrary identity data (e.g., SCSI port names, SCSI device names)
0Ch	ID_FC_NAME	FC-SP certificates that certify a Fibre Channel name as an identity to be used (see RFC 4595 and FC-SP)
all other values	Prohibited	
<sup>a</sup> See RFC 4306 and RFC 4595 for additional information about the associated identification data for these names.		

The contents of the IDENTIFICATION DATA field depend on the value in the ID TYPE field.

When the Certificate payload is included in the parameter data, the identity in the Identification – Application Client payload or Identification – Device Server payload is not required to match anything in the Certificate payload (see

RFC 4306). A mechanism outside the scope of this standard shall be provided to configure any application client or device server to require a match between the identity in an Identification payload and the subject name or subject alternative name in a Certificate payload.

If a device server receives an Identification – Application Client payload that does not conform to the requirements in RFC 4306 or the requirements in this subclause, then the IKEv2-SCSI CCS state maintained for the I\_T\_L nexus shall be abandoned as described in 7.6.3.8.3.

If an application client receives an Identification – Device Server payload that does not conform to the requirements in RFC 4306 or the requirements in this subclause, then the application client should abandon the IKEv2-SCSI CCS and notify the device server that it is abandoning the IKEv2-SCSI CCS as described in 5.14.4.12.

#### 7.6.3.5.5 Certificate payload and Certificate Request payload

The Certificate Request payload (see table 408) allows an application client or device server to request the use of certificates as part of identity authentication and to name one or more trust anchors (see RFC 4306) for the certificate verification process. The Certificate payload (see table 408) delivers a requested identity authentication certificate. The protocol for using Certificate Request payloads and Certificate payloads is described in 5.14.4.3.

**Table 408 — Certificate Request payload and Certificate payload format**

Bit Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT (1b)	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4	CERTIFICATE ENCODING							
5	CERTIFICATE DATA							
n								

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.6.3.5.1.

The CRIT bit is set to one in the Certificate Request payload and the Certificate payload.

The CERTIFICATE ENCODING field describes the contents of the CERTIFICATE DATA field and shall contain one of the values shown in table 409.

**Table 409 — CERTIFICATE ENCODING field**

Code	Description	Reference
00h	Reserved	
01h to 03h	Prohibited	Annex C
04h	X.509 Certificate - Signature	RFC 4306
05h to 0Ah	Prohibited	Annex C
0Bh	Raw RSA Key	RFC 4306 and RFC 4718
0Ch to 0Dh	Prohibited	Annex C
0Eh to C8h	Restricted	IANA
C9h to FFh	Reserved	

The contents of the CERTIFICATE DATA field depend on the value in the CERTIFICATE ENCODING field.

The relationship between the Certificate payload and the Identification payload is described in 7.6.3.5.4.

Device servers that support certificates should support a mechanism outside the scope of this standard for replacing certificates and have the ability to store more than one certificate to facilitate such replacements.

#### 7.6.3.5.6 Authentication payload

The Authentication payload (see table 410) allows the application client and a device server to verify that the data transfers in their IKEv2-SCSI CCS have not be compromised by a man-in-the-middle attack (see 5.14.1.4).

**Table 410 — Authentication payload format**

Bit Byte	7	6	5	4	3	2	1	0	
0	NEXT PAYLOAD								
1	CRIT (1b)	Reserved							
2	(MSB)	IKE PAYLOAD LENGTH (n+1)							(LSB)
3									
4	AUTH METHOD								
5	Reserved								
7									
8	AUTHENTICATION DATA								
n									

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.6.3.5.1.

The CRIT bit is set to one in the Authentication payload.

The AUTH METHOD field indicates the authentication algorithm to be applied to this Authentication payload. The AUTH METHOD field contains the least significant eight bits of the ALGORITHM IDENTIFIER field in an SA\_AUTH\_OUT or an SA\_AUTH\_IN algorithm descriptor (see 7.6.3.6.6) from the Key Exchange step (see 5.14.4.8).

If the contents of the AUTH METHOD field in the parameter data for the Authentication step SECURITY PROTOCOL OUT command (see 5.14.4.9.2) do not match the least significant eight bits in the ALGORITHM IDENTIFIER field in the SA\_AUTH\_OUT algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.6.3.5.12) in the Key Exchange step, then:

- a) The SECURITY PROTOCOL OUT command shall be terminated with CHECK CONDITION status, with the sense key set to ABORTED COMMAND and the additional sense code set to AUTHENTICATION FAILED; and
- b) The device server shall abandon the IKEv2-SCSI CCS (see 5.14.4.12).

If the contents of the AUTH METHOD field in the parameter data for the Authentication step SECURITY PROTOCOL IN command (see 5.14.4.9.3) do not match the least significant eight bits in the ALGORITHM IDENTIFIER field in the SA\_AUTH\_IN algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload in the Key Exchange step, then application client should abandon the IKEv2-SCSI CCS and notify the device server that it is abandoning the IKEv2-SCSI CCS as described in 5.14.4.12.

In the Authentication step SECURITY PROTOCOL OUT command (see 5.14.4.9.2) parameter list, the AUTHENTICATION DATA field contains the result of applying the algorithm specified by the ALGORITHM IDENTIFIER field in the SA\_AUTH\_OUT IKEv2-SCSI cryptographic algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.6.3.5.12) in the Key Exchange step (see 5.14.4.8) as described in table 431 (see 7.6.3.6.6) and this subclause to the following concatenation of bytes:

- 1) All the bytes in the Data-In Buffer returned by the Device Server Capabilities step (see 5.14.4.7) SECURITY PROTOCOL IN command;
- 2) All the bytes in the Data-Out Buffer sent by the Key Exchange step SECURITY PROTOCOL OUT command (see 5.14.4.8.2) in the same IKEv2-SCSI CCS for which GOOD status was returned;
- 3) All the bytes in the IKEv2-SCSI payload-specific data part (see 7.6.3.5.1) of the Nonce payload (see 7.6.3.5.7) that was received in the Key Exchange step SECURITY PROTOCOL IN command in the same IKEv2-SCSI CCS; and
- 4) All the bytes produced by applying the PRF selected by the PRF IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.3) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.6.3.5.12) to the following inputs:
  - 1) The SK\_pi shared key (see 5.14.4.4); and
  - 2) All the bytes in the IKEv2-SCSI payload-specific data part (see 7.6.3.5.1) of the Identification – Application Client payload (see 7.6.3.5.4).

When processing the Authentication step SECURITY PROTOCOL OUT command, the device server shall compute the expected contents of the AUTHENTICATION DATA field by applying the algorithm specified by the ALGORITHM IDENTIFIER field in the SA\_AUTH\_OUT IKEv2-SCSI cryptographic algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.6.3.5.12) in the Key Exchange step (see 5.14.4.8) as described in table 431 (see 7.6.3.6.6) and this subclause to the following concatenation of bytes:

- 1) All the bytes in the Data-In Buffer that the device server returned to any application client in response the last received the Device Server Capabilities step SECURITY PROTOCOL IN command;
- 2) All the bytes in the Data-Out Buffer received in the Key Exchange step SECURITY PROTOCOL OUT command (see 5.14.4.8.2) in the same IKEv2-SCSI CCS for which GOOD status was returned;
- 3) All the bytes in the IKEv2-SCSI payload-specific data part (see 7.6.3.5.1) of the Nonce payload (see 7.6.3.5.7) sent in the Key Exchange step SECURITY PROTOCOL IN command in the same IKEv2-SCSI CCS; and
- 4) All the bytes produced by applying the PRF selected by the PRF IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.3) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.6.3.5.12) to the following inputs:
  - 1) The SK\_pi shared key (see 5.14.4.4); and
  - 2) All the bytes received in the IKEv2-SCSI payload-specific data part (see 7.6.3.5.1) of the Identification – Application Client payload (see 7.6.3.5.4) of the parameter list being processed.

If the expected contents of the AUTHENTICATION DATA field do not match actual contents of the AUTHENTICATION DATA field, then:

- a) The SECURITY PROTOCOL OUT command shall be terminated with CHECK CONDITION status, with the sense key set to ABORTED COMMAND and the additional sense code set to AUTHENTICATION FAILED; and
- b) The device server shall abandon the IKEv2-SCSI CCS (see 5.14.4.12).

For the Authentication step SECURITY PROTOCOL IN command (see 5.14.4.9.3) parameter list, the device server shall compute the AUTHENTICATION DATA field contains by applying the algorithm specified by the ALGORITHM IDENTIFIER field in the SA\_AUTH\_IN IKEv2-SCSI cryptographic algorithm descriptor in the IKEv2-SCSI SA Crypto-

graphic Algorithms payload (see 7.6.3.5.12) in the Key Exchange step (see 5.14.4.8) as described in table 431 (see 7.6.3.6.6) and this subclause to the following concatenation of bytes:

- 1) All the bytes in the Data-In Buffer that the device server returned to any application client in response the last received the Device Server Capabilities step SECURITY PROTOCOL IN command;
- 2) All the bytes in the Data-In Buffer sent by the most recent Key Exchange step SECURITY PROTOCOL IN command (see 5.14.4.8.3) in the same IKEv2-SCSI CCS for which GOOD status was returned;
- 3) All the bytes in the IKEv2-SCSI payload-specific data part (see 7.6.3.5.1) of the Nonce payload (see 7.6.3.5.7) received in the Key Exchange step SECURITY PROTOCOL OUT command in the same IKEv2-SCSI CCS; and
- 4) All the bytes produced by applying the PRF selected by the PRF IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.3) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.6.3.5.12) to the following inputs:
  - 1) The SK<sub>pr</sub> shared key (see 5.14.4.4); and
  - 2) All the bytes in the IKEv2-SCSI payload-specific data part (see 7.6.3.5.1) of the Identification – Device Server payload (see 7.6.3.5.4).

After GOOD status is received for the Authentication step SECURITY PROTOCOL IN command, the application client should compute the expected contents of the AUTHENTICATION DATA field by applying the algorithm specified by the ALGORITHM IDENTIFIER field in the SA\_AUTH\_IN IKEv2-SCSI cryptographic algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.6.3.5.12) in the Key Exchange step (see 5.14.4.8) as described in table 431 (see 7.6.3.6.6) and this subclause to the following concatenation of bytes:

- 1) All the bytes in the Data-In Buffer returned by the Device Server Capabilities step (see 5.14.4.7) SECURITY PROTOCOL IN command;
- 2) All the bytes in the Data-In Buffer returned by the most recent Key Exchange step SECURITY PROTOCOL IN command (see 5.14.4.8.3) in the same IKEv2-SCSI CCS for which GOOD status was returned;
- 3) All the bytes in the IKEv2-SCSI payload-specific data part (see 7.6.3.5.1) of the Nonce payload (see 7.6.3.5.7) sent in the Key Exchange step SECURITY PROTOCOL OUT command in the same IKEv2-SCSI CCS; and
- 4) All the bytes produced by applying the PRF selected by the PRF IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.3) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.6.3.5.12) to the following inputs:
  - 1) The SK<sub>pr</sub> shared key (see 5.14.4.4); and
  - 2) All the bytes in the IKEv2-SCSI payload-specific data part (see 7.6.3.5.1) of the Identification – Device Server payload (see 7.6.3.5.4) received in the SECURITY PROTOCOL IN parameter data.

If the expected contents of the AUTHENTICATION DATA field do not match actual contents of the AUTHENTICATION DATA field, then application client should abandon the IKEv2-SCSI CCS and notify the device server that it is abandoning the IKEv2-SCSI CCS as described in 5.14.4.12.

If the AUTH METHOD field contains 01h (i.e., RSA Digital Signature) the RSA digital signature shall be encoded with the EMSA-PKCS1-v1\_5 signature encoding method as specified in RFC 2437 (see RFC 4718).

### 7.6.3.5.7 Nonce payload

The Nonce payload (see table 411) transfers one random nonce (see 3.1.115) from the application client to the device server or from the device server to the application client.

**Table 411 — Nonce payload format**

Bit Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT (1b)	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4	NONCE DATA							
n								

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.6.3.5.1.

The CRIT bit is set to one in the Nonce payload.

In the Key Exchange step SECURITY PROTOCOL OUT command (see 5.14.4.8.2) the NONCE DATA field contains the application client's random nonce.

In the Key Exchange step SECURITY PROTOCOL IN command (see 5.14.4.8.3) the NONCE DATA field contains the device server's random nonce.

The requirements that RFC 4306 places on the nonce data shall apply to this standard.

### 7.6.3.5.8 Notify payload

This standard uses the Notify payload (see table 412) only to provide initial contact notification from the application client to the device server.

**Table 412 — Notify payload format**

Bit Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT (1b)	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (0010h)						
3								(LSB)
4	PROTOCOL ID (01h)							
5	SAI SIZE (08h)							
6	(MSB)	NOTIFY MESSAGE TYPE (4000h)						
7								(LSB)
8	Restricted (see RFC 4306)							
11								
12	(MSB)	SAI						
15								(LSB)

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.6.3.5.1.

The CRIT bit is set to one in the Notify payload.

The PROTOCOL ID field contains one. If the device server receives a PROTOCOL ID field set to a value other than one, then the IKEv2-SCSI CCS state maintained for the I\_T\_L nexus shall be abandoned as described in 7.6.3.8.3.

The SAI SIZE field contains eight. If the device server receives a value other than eight in the SAI SIZE field, then the IKEv2-SCSI CCS state maintained for the I\_T\_L nexus shall be abandoned as described in 7.6.3.8.3.

The NOTIFY MESSAGE TYPE field contains 16 384 (i.e., INITIAL\_CONTACT). If the device server receives a value other than 16 384 in the NOTIFY MESSAGE TYPE field, then the IKEv2-SCSI CCS state maintained for the I\_T\_L nexus shall be abandoned as described in 7.6.3.8.3.

The SAI field contains the device server's SAI. If the contents of the SAI field are not identical to the contents of the IKE\_SA DEVICE SERVER SAI field in the IKEv2-SCSI header (see 7.6.3.4), then the IKEv2-SCSI CCS state maintained for the I\_T\_L nexus shall be abandoned as described in 7.6.3.8.3.

Unless an error is detected, the device server shall process the Notify payload as described in 5.14.4.9.2.



### 7.6.3.5.9 Delete payload

The Delete payload (see table 413) requests the deletion of an existing SA or the abandonment of an IKEv2-SCSI CCS that is in progress.

**Table 413 — Delete payload format**

Bit Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT (1b)	Reserved						
2	(MSB)							
3	IKE PAYLOAD LENGTH (0018h)							(LSB)
4	PROTOCOL ID (01h)							
5	SAI SIZE (08h)							
6	(MSB)							
7	NUMBER OF SAIS (0002h)							(LSB)
8								
11	Restricted (see RFC 4306)							
12	(MSB)							
15	AC_SAI							(LSB)
16								
19	Restricted (see RFC 4306)							
20	(MSB)							
23	DS_SAI							(LSB)

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.6.3.5.1.

The CRIT bit is set to one in the Delete payload.

The PROTOCOL ID field contains one. If the device server receives a PROTOCOL ID field set to a value other than one, then the error shall be processed as described in 7.6.3.8.3.

The SAI SIZE field contains eight. If the device server receives a value other than eight in the SAI SIZE field, then the error shall be processed as described in 7.6.3.8.3.

The AC\_SAI field contains the AC\_SAI SA parameter value (see 3.1.126) for the SA to be deleted. If the contents of the AC\_SAI field do not match the contents of the IKE\_SA APPLICATION CLIENT SAI field in the IKEv2-SCSI header (see 7.6.3.4), then the error shall be processed as described in 7.6.3.8.3.

The DS\_SAI field contains the DS\_SAI SA parameter value (see 3.1.126) for the SA to be deleted. If the contents of the DS\_SAI field do not match the contents of the IKE\_SA DEVICE SERVER SAI field in the IKEv2-SCSI header (see 7.6.3.4), then the error shall be processed as described in 7.6.3.8.3.

If the device server is maintaining SA parameters for which the AC\_SAI matches the contents of the AC\_SAI field and the DS\_SAI matches the contents of the DS\_SAI field, that set of SA parameters shall be deleted.

If the device server is maintaining state for an IKEv2-SCSI CCS on the I\_T\_L nexus on which the command was received, the IKEv2-SCSI CCS shall be abandoned (see 5.14.4.12) if:

- a) The contents of the AC\_SAI field match the application client's SAI in the maintained state; and
- b) The contents of the DS\_SAI field match the device server's SAI in the maintained state.

#### **7.6.3.5.10 Encrypted payload**

##### **7.6.3.5.10.1 Combined mode encryption**

The following types of encryption algorithms are used in the Encrypted payload:

- a) Non-combined modes that use separate algorithms to encrypt/decrypt and integrity check the Encrypted payload; and
- b) Combined modes in which a single encryption algorithm does both encryption/decryption and integrity checking.

The ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.4) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.6.3.5.12) in the Key Exchange step (see 5.14.4.8) indicates the type of encryption algorithm to be applied to the Encrypted payload for IKEv2-SCSI functions as follows:

- a) If the ALGORITHM IDENTIFIER field is not set to AUTH\_COMBINED, a non-combined mode encryption algorithm is being used; or
- b) If the ALGORITHM IDENTIFIER field is set to AUTH\_COMBINED, a combined mode encryption algorithm is being used.

If the Encrypted payload is in parameter data that is not associated with an active IKEv2-SCSI CCS (see 3.1.66), then the integrity checking algorithm identifier that selects between combined and non-combined mode encryption is found in the MGMT\_DATA SA parameter (see 3.1.126 and 5.14.4.11).

### 7.6.3.5.10.2 Encrypted payload introduction

The Encrypted payload transfers (see table 414) one or more other IKEv2-SCSI payloads that are encrypted and integrity checked from the application client to the device server and vice versa.

If IKEv2-SCSI parameter data contains the Encrypted payload, then the Encrypted payload is the first payload in the parameter data (i.e., the NEXT PAYLOAD field in the IKEv2-SCSI header (see 7.6.3.5.1) contains 2Eh). Since the NEXT PAYLOAD field in an Encrypted payload identifies the first payload in the CIPHERTEXT field, there is no way to identify a payload following the Encrypted payload, and none are allowed.

**Table 414 — Encrypted payload format**

Bit Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT (1b)	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								
4	INITIALIZATION VECTOR							
i-1								
i								
k-1	CIPHERTEXT							
k								
n								
	INTEGRITY CHECK VALUE							

The NEXT PAYLOAD field identifies the first IKEv2-SCSI payload in the CIPHERTEXT field using the coded values shown in table 401 (see 7.6.3.5.1).

The CRIT bit and IKE PAYLOAD LENGTH field are described in 7.6.3.5.1.

The CRIT bit is set to one in the Encrypted payload.

The IKE PAYLOAD LENGTH field contains the number of bytes that follow in the Encrypted payload. The number of bytes in the CIPHERTEXT field is equal to the number of bytes in the plaintext (see table 415).

The INITIALIZATION VECTOR field contains the initialization vector encryption algorithm input value. The size of the initialization vector is defined by the encryption algorithm as shown in table 423 (see 7.6.3.6.2).

The CIPHERTEXT field contains an output of the encryption algorithm specified by the ALGORITHM IDENTIFIER field in the ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.2) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.6.3.5.12) in the Key Exchange step (see 5.14.4.8) using the inputs:

- a) IKEv2-SCSI AAD is an encryption algorithm input as follows:
  - A) If a non-combined mode encryption algorithm is being used (see 7.6.3.5.10.1), then no AAD input is needed or provided; or
  - B) If a combined mode encryption algorithm is being used (see 7.6.3.5.10.1), then the AAD described in 7.6.3.5.10.3 is an input;
- b) The contents of the INITIALIZATION VECTOR field (see table 423);
- c) The plaintext data shown in table 415; and

- d) The shared key value, key length, and salt value (see table 423 in 7.6.3.6.2) if any, for one of the following shared keys:
  - A) If the Encrypted payload appears in the parameter data for an Authentication step SECURITY PROTOCOL OUT command (see 5.14.4.9.2) or the parameter data for a Delete operation SECURITY PROTOCOL OUT command (see 5.14.4.13) that affects an active IKEv2-SCSI CCS (see 3.1.66), then the SK\_ei shared key (see 5.14.4.4) for the IKEv2-SCSI CCS;
  - B) If the Encrypted payload appears in the parameter data for an Authentication step SECURITY PROTOCOL IN command (see 5.14.4.9.3), then the SK\_er shared key (see 5.14.4.4) for the IKEv2-SCSI CCS; or
  - C) If the Encrypted payload appears in the parameter data for a Delete operation SECURITY PROTOCOL OUT command (see 5.14.4.13) that does not affect an active IKEv2-SCSI CCS, then the SK\_ei shared key (see 5.14.4.4) from the MGMT\_DATA SA parameter (see 3.1.126 and 5.14.4.11).

NOTE 68 - Salt values (see table 423 in 7.6.3.6.2) are used only by combined mode encryption algorithms (see 7.6.3.5.10.1).

If the Encrypted payload appears in the parameter data for a Delete operation SECURITY PROTOCOL OUT command that does not affect an active IKEv2-SCSI CCS, then the encryption algorithm identifier stored in the MGMT\_DATA SA parameter indicates the encryption algorithm to use.

The INTEGRITY CHECK VALUE field contains the integrity check value that is computed as described in 7.6.3.5.10.1 (e.g., if the integrity algorithm is AUTH\_COMBINED, then the integrity check value is an output of the encryption algorithm). The size of the integrity check value is defined by the integrity algorithm or encryption algorithm, depending on which algorithm computes the value.

If the integrity algorithm specified by the ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload in the Key Exchange step is not AUTH\_COMBINED, then the contents of the INTEGRITY CHECK VALUE field are computed by processing the integrity check algorithm specified by the ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor using the following inputs:

- a) A byte string composed of:
  - 1) The AAD described in 7.6.3.5.10.3;
  - 2) The contents of the INITIALIZATION VECTOR field (see table 414); and
  - 3) The ciphertext that is the result of encrypting the plaintext data;
 and
- b) The shared key value from one of the following shared keys:
  - A) If the Encrypted payload appears in the parameter data for an Authentication step SECURITY PROTOCOL OUT command (see 5.14.4.9.2) or the parameter data for a Delete operation SECURITY PROTOCOL OUT command (see 5.14.4.13) that affects an active IKEv2-SCSI CCS (see 3.1.66), then the SK\_ai shared key (see 5.14.4.4) for the IKEv2-SCSI CCS;
  - B) If the Encrypted payload appears in the parameter data for an Authentication step SECURITY PROTOCOL IN command (see 5.14.4.9.3), then the SK\_ar shared key (see 5.14.4.4) for the IKEv2-SCSI CCS; or
  - C) If the Encrypted payload appears in the parameter data for a Delete operation SECURITY PROTOCOL OUT command (see 5.14.4.13) that does not affect an active IKEv2-SCSI CCS, then the SK\_ai shared key (see 5.14.4.4) from the MGMT\_DATA SA parameter (see 3.1.126 and 5.14.4.11).

If the Encrypted payload appears in the parameter data for a Delete operation SECURITY PROTOCOL OUT command that does not affect an active IKEv2-SCSI CCS, then the integrity checking algorithm identifier stored in the MGMT\_DATA SA parameter indicates the integrity checking algorithm to use.

When computing the encrypted CIPHERTEXT field contents for an Encrypted payload, the plaintext shown in table 415 is used.

**Table 415 — Plaintext format for Encrypted payload CIPHERTEXT field**

Bit Byte	7	6	5	4	3	2	1	0
IKEv2-SCSI payloads								
0	IKEv2-SCSI payload [first] (see 7.6.3.5)							
	⋮							
n	IKEv2-SCSI payload [last] (see 7.6.3.5)							
p	PADDING BYTES (if any)							
k-1								
k	PAD LENGTH (k-p)							

Each IKEv2-SCSI payload (see 7.6.3.5) contains specific data related to the operation being performed. A specific combination of IKEv2-SCSI payloads is needed for each operation (e.g., Authentication) as summarized in 5.14.4.2.

The PADDING BYTES field contains zero to 255 bytes. The number of padding bytes is:

- a) Defined by the encryption algorithm; or
- b) Any number of bytes that causes the length of all plaintext bytes (i.e.,  $l+2$ ) to be a whole multiple of the alignment (see table 423 in 7.6.3.6.2) for the encryption algorithm being used.

The contents of the padding bytes are:

- a) Defined by the encryption algorithm; or
- b) If the encryption algorithm does not define the padding bytes contents, a series of one byte binary values starting at one and incrementing by one in each successive byte (i.e., 01h in the first padding byte, 02h in the second padding byte, etc.).

If the encryption algorithm does not place requirements on the contents of the padding bytes (i.e., option b) is in effect), then after decryption the contents of the padding bytes shall be verified to match the series of one byte binary values described in this subclause. If this verification is not successful in a device server, the error shall be processed as described in 7.6.3.8.2. If this verification is not successful in an application client, the decrypted data should be ignored.

The PAD LENGTH field contains the number of bytes in the PADDING BYTES field.

#### 7.6.3.5.10.3 IKEv2-SCSI AAD

The AAD defined by this standard for IKEv2-SCSI use is as follows:

- 1) All the bytes in the IKEv2-SCSI Header (see 7.6.3.4); and
- 2) All the bytes in the IKEv2-SCSI Payload Header (see 7.6.3.5.1) of the Encrypted payload (see 7.6.3.5.10).

#### 7.6.3.5.10.4 Processing a received Encrypted payload

Before performing any checks of data contained in the CIPHERTEXT field, the contents of the INTEGRITY CHECK VALUE field and CIPHERTEXT field shall be integrity checked and decrypted based on the contents of the IKE\_SA APPLICATION CLIENT SAI field and the IKE\_SA DEVICE SERVER SAI field in the IKEv2-SCSI header (see 7.6.3.4) as described in this subclause.

Computation of the comparison integrity check value and decryption of an Encrypted payload is performed as follows:

- 1) If a non-combined mode encryption algorithm is being used (see 7.6.3.5.10.1), then the comparison integrity check value is computed by performing the integrity check algorithm specified by the ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor using the following inputs:
  - A) A byte string composed of:
    - 1) The AAD described in 7.6.3.5.10.3;
    - 2) The contents of the INITIALIZATION VECTOR field (see table 414) in the Encrypted payload; and
    - 3) The contents of the ciphertext field (see table 414) in the Encrypted payload;
 and
  - B) The shared key value from one of the following shared keys:
    - a) If the Encrypted payload appears in the parameter data for an Authentication step SECURITY PROTOCOL OUT command (see 5.14.4.9.2) or the parameter data for a Delete operation SECURITY PROTOCOL OUT command (see 5.14.4.13) that affects an active IKEv2-SCSI CCS (see 3.1.66), then the SK\_ai shared key (see 5.14.4.4) for the IKEv2-SCSI CCS;
    - b) If the Encrypted payload appears in the parameter data for an Authentication step SECURITY PROTOCOL IN command (see 5.14.4.9.3), then the SK\_ar shared key (see 5.14.4.4) for the IKEv2-SCSI CCS; or
    - c) If the Encrypted payload appears in the parameter data for a Delete operation SECURITY PROTOCOL OUT command (see 5.14.4.13) that does not affect an active IKEv2-SCSI CCS, then the SK\_ai shared key (see 5.14.4.4) from the MGMT\_DATA SA parameter (see 3.1.126 and 5.14.4.11);
 and
- 2) Decrypt the CIPHERTEXT field to produce plaintext data – and if a combined mode encryption algorithm is being used (see 7.6.3.5.10.1), produce the comparison integrity check value – using the encryption algorithm specified by the ALGORITHM IDENTIFIER field in the ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.2) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.6.3.5.12) in the Key Exchange step (see 5.14.4.8) using the inputs:
  - A) IKEv2-SCSI AAD is an input to the encryption algorithm as follows:
    - a) If a non-combined mode encryption algorithm is being used (see 7.6.3.5.10.1), then no AAD input is needed or provided; or
    - b) If a combined mode encryption algorithm is being used (see 7.6.3.5.10.1), then the AAD described in 7.6.3.5.10.3 is an input;
  - B) The contents of the INITIALIZATION VECTOR field (see table 414) in the Encrypted payload;
  - C) The contents of the ciphertext field (see table 414) in the Encrypted payload; and
  - D) The shared key value, key length, and salt value (see table 423 in 7.6.3.6.2) if any, for one of the following shared keys:
    - a) If the Encrypted payload appears in the parameter data for an Authentication step SECURITY PROTOCOL OUT command (see 5.14.4.9.2), or the parameter data for a Delete operation SECURITY PROTOCOL OUT command (see 5.14.4.13) that affects an active IKEv2-SCSI CCS (see 3.1.66), then the SK\_ei shared key (see 5.14.4.4) for the IKEv2-SCSI CCS;
    - b) If the Encrypted payload appears in the parameter data for an Authentication step SECURITY PROTOCOL IN command (see 5.14.4.9.3), then the SK\_er shared key (see 5.14.4.4) for the IKEv2-SCSI CCS; or
    - c) If the Encrypted payload appears in the parameter data for a Delete operation SECURITY PROTOCOL OUT command (see 5.14.4.13) that does not affect an active IKEv2-SCSI CCS, then

the SK\_ei shared key (see 5.14.4.4) from the MGMT\_DATA SA parameter (see 3.1.126 and 5.14.4.11).

If the Encrypted payload appears in the parameter data for a Delete operation SECURITY PROTOCOL OUT command that does not affect an active IKEv2-SCSI CCS, then the integrity checking algorithm identifier value and encryption algorithm identifier value that are stored in the MGMT\_DATA SA parameter indicate the integrity checking algorithm to use.

If the comparison integrity check value differs from the value in the INTEGRITY CHECK VALUE field of the Encrypted payload:

- a) The application client should abandon the IKEv2-SCSI CCS and notify the device server that it is abandoning the IKEv2-SCSI CCS as described in 5.14.4.12; and
- b) The device server shall respond to the mismatch as follows:
  - A) If the IKEv2-SCSI header (see 7.6.3.4) specifies an attempt to provide authentication data for or the deletion of an IKE-v2-SCSI CCS on the I\_T\_L nexus on which the command was received, then:
    - a) The SECURITY PROTOCOL OUT command shall be terminated with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to SA CREATION PARAMETER VALUE REJECTED; and
    - b) The device server shall continue the IKEv2-SCSI CCS by preparing to receive another Authentication step SECURITY PROTOCOL OUT command;
  - or
  - B) If the IKEv2-SCSI header (see 7.6.3.4) specifies the deletion of an active SA, then the SECURITY PROTOCOL OUT command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

7.6.3.5.11 IKEv2-SCSI SA Creation Capabilities payload

The IKEv2-SCSI SA Creation Capabilities payload (see table 416) lists all the security algorithms that the device server allows to be used in an IKEv2-SCSI CCS. Events that are outside the scope of this standard may change the contents of the IKEv2-SCSI SA Creation Capabilities payload at any time.

Table 416 — IKEv2-SCSI SA Creation Capabilities payload format

Bit Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD (00h)							
1	CRIT (1b)	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4	Reserved							
6								
7	NUMBER OF ALGORITHM DESCRIPTORS							
IKEv2-SCSI cryptographic algorithm descriptors								
8	IKEv2-SCSI cryptographic algorithm descriptor [first] (see 7.6.3.6)							
	⋮							
	IKEv2-SCSI cryptographic algorithm descriptor [last] (see 7.6.3.6)							
n								

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.6.3.5.1.

The NEXT PAYLOAD field is set to zero (i.e., No Next Payload) in the IKEv2-SCSI SA Creation Capabilities payload.

The CRIT bit is set to one in the IKEv2-SCSI SA Creation Capabilities payload.

The NUMBER OF ALGORITHM DESCRIPTORS field contains the number of IKEv2-SCSI cryptographic algorithm descriptors that follow in the IKEv2-SCSI SA Creation Capabilities payload.

Each IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6) describes one combination of security algorithm and algorithm attributes that the device server allows to be used in an IKEv2-SCSI CCS. If more than one set of algorithm attributes (e.g., key length) is allowed for any allowed security algorithm, a different SCSI cryptographic algorithms descriptor shall be included for each set of algorithm attributes.

The SCSI cryptographic algorithms descriptors shall be ordered by:

- 1) Increasing algorithm type;
- 2) Increasing algorithm identifier within the same algorithm type; and
- 3) Increasing key length, if any, within the same algorithm identifier.

The algorithms allowed may be a subset of the algorithms supported by the device server.



The method for changing which of the device server supported algorithms are allowed is outside the scope of this standard, but changes in allowed algorithms do not take effect until the new list is returned to any application client in an IKEv2-SCSI SA Creation Capabilities payload.

#### 7.6.3.5.12 IKEv2-SCSI SA Cryptographic Algorithms payload

The IKEv2-SCSI SA Cryptographic Algorithms payload (see table 417) lists the security algorithms that are being used in the creation and management (e.g., deletion) of an SA using an IKEv2-SCSI CCS.

**Table 417 — IKEv2-SCSI SA Cryptographic Algorithms payload format**

Bit Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT (1b)	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4	Reserved							
13								
14	(MSB)	USAGE DATA LENGTH (0000h)						
15								(LSB)
16	Reserved							
18								
19	NUMBER OF ALGORITHM DESCRIPTORS							
IKEv2-SCSI cryptographic algorithm descriptors								
20	IKEv2-SCSI cryptographic algorithm descriptor [first] (see 7.6.3.6)							
	⋮							
	IKEv2-SCSI cryptographic algorithm descriptor [last] (see 7.6.3.6)							
n								

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.6.3.5.1.

The CRIT bit is set to one in the IKEv2-SCSI SA Cryptographic Algorithms payload.

The USAGE DATA LENGTH field is set to zero in the IKEv2-SCSI SA Cryptographic Algorithms payload.

The NUMBER OF ALGORITHM DESCRIPTORS field contains the number of IKEv2-SCSI cryptographic algorithm descriptors that follow in the IKEv2-SCSI SA Cryptographic Algorithms payload. If a device server receives a NUMBER OF ALGORITHM DESCRIPTORS field that does not contain six, then the error shall be processed as described in 7.6.3.8.3.

Each IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6) describes one combination of security algorithm and algorithm attributes to be used during the IKEv2-SCSI CCS.

The IKEv2-SCSI cryptographic algorithm descriptors are ordered as follows:

- 1) One ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.2);
- 2) One PRF IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.3);
- 3) One INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.4);
- 4) One D-H IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.5);
- 5) One SA\_AUTH\_OUT IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.6).
- 6) One SA\_AUTH\_IN IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.6).

If a device server receives an IKEv2-SCSI SA Cryptographic Algorithms payload that does not contain the IKEv2-SCSI cryptographic algorithm descriptors described in this subclause in the order described in this subclause, then the error shall be processed as described in 7.6.3.8.3.

If the device server receives an IKEv2-SCSI SA Cryptographic Algorithms payload that contains an ENCR IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to ENCR\_NULL, then the error shall be processed as described in 7.6.3.8.3.

In the Key Exchange step SECURITY PROTOCOL IN parameter data (see 5.14.4.8.3), the device server returns the IKEv2-SCSI SA Cryptographic Algorithms payload received during the Key Exchange step SECURITY PROTOCOL OUT command (see 5.14.4.8.2) to confirm acceptance of the algorithms.

### 7.6.3.5.13 IKEv2-SCSI SAUT Cryptographic Algorithms payload

The IKEv2-SCSI SAUT Cryptographic Algorithms payload (see table 418) lists the usage type of and security algorithms to be used by the SA that is created as a result of an IKEv2-SCSI CCS.

**Table 418 — IKEv2-SCSI SAUT Cryptographic Algorithms payload format**

Bit Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT (1b)	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4	Reserved							
11								
12	(MSB)	SA TYPE						
13								(LSB)
14	(MSB)	USAGE DATA LENGTH (j)						
15								(LSB)
16	USAGE DATA							
16+j-1								
16+j	Reserved							
16+j+2								
16+j+3	NUMBER OF ALGORITHM DESCRIPTORS							
IKEv2-SCSI cryptographic algorithm descriptors								
16+j+4	IKEv2-SCSI cryptographic algorithm descriptor [first] (see 7.6.3.6)							
	⋮							
	IKEv2-SCSI cryptographic algorithm descriptor [last] (see 7.6.3.6)							
n								

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.6.3.5.1.

The CRIT bit is set to one in the IKEv2-SCSI SAUT Cryptographic Algorithms payload.

The SA TYPE field specifies the usage type for the SA and is selected from among those listed in table 58 (see 5.14.2.2). An error shall be processed as described in 7.6.3.8.3 if any of the following conditions occur:

- The device server receives an SA usage type whose use the device server does not allow;
- An SA usage type between 8000h and BFFFh inclusive is received in a Key Exchange step SECURITY PROTOCOL OUT command (see 5.14.4.8.2); or
- An SA usage type between A000h and CFFFh inclusive is received for which the device server is unable to verify the applicable usage type constraints.

The method for changing which of the device server supported SA usage types are allowed is outside the scope of this standard.

The USAGE DATA LENGTH field specifies number of bytes of usage data that follow.

The size and format of the usage data depends on the SA type (see table 58 in 5.14.2.2). If the device server receives a USAGE DATA LENGTH field that contains a value that is inconsistent with the SA type, then the error shall be processed as described in 7.6.3.8.3.

The USAGE DATA field contains information to be stored in the USAGE\_DATA SA parameter (see 3.1.126) if the SA is generated (see 5.14.4.11).

The NUMBER OF ALGORITHM DESCRIPTORS field contains the number of IKEv2-SCSI cryptographic algorithm descriptors that follow in the IKEv2-SCSI SAUT Cryptographic Algorithms payload. If a device server receives a NUMBER OF ALGORITHM DESCRIPTORS field that contains a value other than two, then the error shall be processed as described in 7.6.3.8.3.

Each IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6) describes one combination of security algorithm and algorithm attributes to be used by the SA created as a result of the IKEv2-SCSI CCS. The IKEv2-SCSI cryptographic algorithm descriptors are ordered as follows:

- 1) One ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.2); and
- 2) One INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.4).

If a device server receives an IKEv2-SCSI SAUT Cryptographic Algorithms payload that does not contain the IKEv2-SCSI cryptographic algorithm descriptors described in this subclause in the order described in this subclause, then the error shall be processed as described in 7.6.3.8.3.

#### 7.6.3.5.14 IKEv2-SCSI Timeout Values payload

The IKEv2-SCSI Timeout Values payload (see table 419) specifies the timeout intervals associated with an IKEv2-SCSI CCS

**Table 419 — IKEv2-SCSI Timeout Values payload format**

Bit Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT (1b)	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (0010h)						
3								(LSB)
4	Reserved							
6								
7	NUMBER OF TIMEOUT VALUES (02h)							
8	(MSB)	IKEV2-SCSI PROTOCOL TIMEOUT						
11								(LSB)
12	(MSB)	IKEV2-SCSI SA INACTIVITY TIMEOUT						
15								(LSB)

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.6.3.5.1.

The CRIT bit is set to one in the IKEv2-SCSI Timeout Values payload.

The NUMBER OF TIMEOUT VALUES field specifies the number of four-byte timeout values that follow. If the number of timeout values is less than two, then the IKEv2-SCSI CCS state maintained for the I\_T\_L nexus shall be abandoned as described in 7.6.3.8.3.

The IKEV2-SCSI PROTOCOL TIMEOUT field specifies the number of seconds that the device server shall wait for the next command in the IKEv2-SCSI CCS. If the timeout expires before the device server receives an IKEv2-SCSI CCS command, the device server shall abandon the IKEv2-SCSI CCS as described in 5.14.4.12.

The IKEV2-SCSI SA INACTIVITY TIMEOUT field specifies the number of seconds that the device server shall wait for the next command that uses an SA. This value is copied to the TIMEOUT SA parameter when the SA is generated (see 5.14.4.11).

The device server shall replace any timeout value that is set to zero with a value of ten (i.e., ten seconds).

The maximum value for the protocol timeout should be long enough to allow the application client to continue the IKEv2-SCSI CCS, but short enough that if an incomplete IKEv2-SCSI CCS is abandoned, the device server discards the state for that IKEv2-SCSI CCS and becomes available to for another IKEv2-SCSI CCS without excessive delay.

7.6.3.6 IKEv2-SCSI cryptographic algorithm descriptors

7.6.3.6.1 Overview

Each IKEv2-SCSI cryptographic algorithm descriptor (see table 420) specifies one algorithm used for encryption, integrity checking, key generation, or authentication.

Table 420 — IKEv2-SCSI cryptographic algorithm descriptor format

Bit Byte	7	6	5	4	3	2	1	0
0	ALGORITHM TYPE							
1	Reserved							
2	(MSB)							
3	IKE DESCRIPTOR LENGTH (000Ch)							(LSB)
4	(MSB)							
7	ALGORITHM IDENTIFIER							(LSB)
8								
11	ALGORITHM ATTRIBUTES							

The ALGORITHM TYPE field (see table 421) specifies the type of cryptographic algorithm to which the IKEv2-SCSI cryptographic algorithm descriptor applies.

**Table 421 — ALGORITHM TYPE field**

Code	Name	Description	Reference
01h	ENCR	Encryption algorithm	7.6.3.6.2
02h	PRF	Pseudo-random function	7.6.3.6.3
03h	INTEG	Integrity algorithm	7.6.3.6.4
04h	D-H	Diffie-Hellman group	7.6.3.6.5
05h to F0h	Restricted		RFC 4306
F9h	SA_AUTH_OUT	IKEv2-SCSI authentication algorithm for SECURITY PROTOCOL OUT data	7.6.3.6.6
FAh	SA_AUTH_IN	IKEv2-SCSI authentication algorithm for SECURITY PROTOCOL IN data	7.6.3.6.6
All others	Reserved		

The IKE DESCRIPTOR LENGTH field contains 12 (i.e., the total number of bytes in the IKEv2-SCSI SA Cryptographic Algorithms descriptor including the ALGORITHM TYPE field and reserved byte).

NOTE 69 - The contents of the IKE DESCRIPTOR LENGTH field differ from those found in most SCSI length fields, however, they are consistent with the IKEv2 usage (see RFC 4306).

The contents of the ALGORITHM IDENTIFIER field and ALGORITHM ATTRIBUTES field depend on the contents of the ALGORITHM TYPE field (see table 421). The ALGORITHM ATTRIBUTES field is reserved in some IKEv2-SCSI SA Cryptographic Algorithms descriptor formats.

### 7.6.3.6.2 Encryption algorithm (ENCR) IKEv2-SCSI cryptographic algorithm descriptor

When the ALGORITHM TYPE field is set to ENCR (i.e., 01h) in an IKEv2-SCSI cryptographic algorithm descriptor (see table 422), the descriptor specifies an encryption algorithm to be applied during the IKEv2-SCSI Authentication step (see 5.14.4.9), SA deletion operation (see 5.14.4.13), and when the SA created by the IKEv2-SCSI is applied to user data.

**Table 422 — ENCR IKEv2-SCSI cryptographic algorithm descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	ALGORITHM TYPE (01h)							
1	Reserved							
2	(MSB)	IKE DESCRIPTOR LENGTH (000Ch)						
3								(LSB)
4	(MSB)	ALGORITHM IDENTIFIER						
7								(LSB)
8		Reserved						
9								
10	(MSB)	KEY LENGTH						
11								(LSB)

The ALGORITHM TYPE field and IKE DESCRIPTOR LENGTH field are described in 7.6.3.6.1.

The ALGORITHM IDENTIFIER field (see table 423) specifies the encryption algorithm to which the ENCR IKEv2-SCSI cryptographic algorithm descriptor applies.

**Table 423 — ENCR ALGORITHM IDENTIFIER field**

Code	Description	Salt <sup>a</sup> length (bytes)	IV <sup>b</sup> length (bytes)	Align- ment <sup>c</sup> (bytes)	Key length (bytes)	Support	Reference
8001 000Bh	ENCR_NULL <sup>d</sup>	n/a	0	4	0	Mandatory	
8001 000Ch	AES-CBC <sup>d</sup>	n/a	16	16	16	Optional	RFC 3602
					24	Prohibited	
					32	Optional	
8001 0010h	AES-CCM with a 16 byte MAC <sup>e</sup>	3	8	4	16	Optional	RFC 4309
					24	Prohibited	
					32	Optional	
8001 0014h	AES-GCM with a 16 byte MAC <sup>e</sup>	4	8	4	16	Optional	RFC 4106
					24	Prohibited	
					32	Optional	
8001 0400h to 8001 FFFFh	Vendor Specific						
0000 0000h to 0000 FFFFh	Restricted						IANA
All other values	Reserved						

<sup>a</sup> See RFC 4106 and RFC 4309.

<sup>b</sup> Initialization Vector.

<sup>c</sup> The alignment required in the plaintext prior to encryption.

<sup>d</sup> If the INTEG cryptographic algorithm descriptor (see 7.6.3.6.4) in the same IKEv2-SCSI SA Cryptographic Algorithms payload or the same IKEv2-SCSI SAUT Cryptographic Algorithms payload as this ENCR cryptographic algorithm descriptor has the ALGORITHM IDENTIFIER field set to AUTH\_COMBINED, then the error shall be processed as described in 7.6.3.8.3.

<sup>e</sup> If the INTEG cryptographic algorithm descriptor (see 7.6.3.6.4) in the same IKEv2-SCSI SA Cryptographic Algorithms payload or the same IKEv2-SCSI SAUT Cryptographic Algorithms payload as this ENCR cryptographic algorithm descriptor does not have the ALGORITHM IDENTIFIER field set to AUTH\_COMBINED, then the error shall be processed as described in 7.6.3.8.3.

ENCR\_NULL indicates that encryption is not to be applied when the SA created by the IKEv2-SCSI is applied to user data.

The IKEv2-SCSI CCS state maintained for the I\_T\_L nexus shall be abandoned as described in 7.6.3.8.3 if the parameter list contains an IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.6.3.5.12) that contains:

- a) An ENCR IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to ENCR\_NULL;
- b) The following combination of IKEv2-SCSI cryptographic algorithm descriptors (see 7.6.3.6.4):
  - A) An INTEG IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to a value other than AUTH\_COMBINED; and



- B) An ENCR IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to a value that table 423 describes as requiring AUTH\_COMBINED as the integrity check algorithm;
- or
- c) The following combination of IKEv2-SCSI cryptographic algorithm descriptors:
  - A) An INTEG IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to AUTH\_COMBINED; and
  - B) An ENCR IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to a value that table 423 does not describe as requiring AUTH\_COMBINED as the integrity check algorithm.

The KEY LENGTH field specifies the number of bytes in the shared key (see 3.1.155) for the encryption algorithm to which the ENCR IKEv2-SCSI cryptographic algorithm descriptor applies.

The IKEv2-SCSI CCS state maintained for the I\_T\_L nexus shall be abandoned as described in 7.6.3.8.3 if the parameter list contains:

- a) No ENCR IKEv2-SCSI cryptographic algorithm descriptors;
- b) More than one ENCR IKEv2-SCSI cryptographic algorithm descriptor;
- c) An ENCR IKEv2-SCSI cryptographic algorithm descriptor that does not appear in the SA Creations Capabilities payload (see 7.6.3.5.11) last returned by the device server to any application client (i.e., an ENCR IKEv2-SCSI cryptographic algorithm descriptor for a combination of algorithm identifier and key length that the device server has not reported as one of its SA creation capabilities); or
- d) An ENCR IKEv2-SCSI cryptographic algorithm descriptor that contains:
  - A) An algorithm identifier that is not shown in table 423; or
  - B) A key length that:
    - a) Does not match one of the values shown in table 423; or
    - b) Is not supported by the device server.

#### 7.6.3.6.3 Pseudo-random function (PRF) IKEv2-SCSI cryptographic algorithm descriptor

When the algorithm type field is set to PRF (i.e., 02h) in an IKEv2-SCSI cryptographic algorithm descriptor (see table 424), the descriptor specifies the pseudo-random function and KDF to be used during the Key Exchange step completion (see 5.14.4.8.4).

**Table 424 — PRF IKEv2-SCSI cryptographic algorithm descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	ALGORITHM TYPE (02h)							
1	Reserved							
2	(MSB) _____							
3	IKE DESCRIPTOR LENGTH (000Ch) _____ (LSB)							
4	(MSB) _____							
7	ALGORITHM IDENTIFIER _____ (LSB)							
8	_____							
11	Reserved _____							

The ALGORITHM TYPE field and IKE DESCRIPTOR LENGTH field are described in 7.6.3.6.1.

The ALGORITHM IDENTIFIER field (see table 425) specifies PRF and KDF to which the PRF IKEv2-SCSI cryptographic algorithm descriptor applies.

**Table 425 — PRF ALGORITHM IDENTIFIER field**

Code	Description	Support	Output length (bytes)	Reference	
				PRF <sup>a</sup>	KDF <sup>b</sup>
8002 0002h	IKEv2-use based on SHA-1	Optional	20	RFC 2104	5.14.3.3
8002 0004h	IKEv2-use based on AES-128 in CBC mode	Optional	16	RFC 4434	5.14.3.4
8002 0005h	IKEv2-use based on SHA-256	Optional	32	RFC 4868	5.14.3.3
8002 0007h	IKEv2-use based on SHA-512	Optional	64	RFC 4868	5.14.3.3
8002 0400h to 8002 FFFFh	Vendor Specific				
0000 0000h to 0000 FFFFh	Restricted			IANA	
All others	Reserved				
<sup>a</sup> PRFs are equivalent to the prf() functions defined in RFC 4306.					
<sup>b</sup> KDFs are equivalent to the prf+() functions defined in RFC 4306.					

The IKEv2-SCSI CCS state maintained for the I\_T\_L nexus shall be abandoned as described in 7.6.3.8.3 if the parameter list contains:

- a) No PRF IKEv2-SCSI cryptographic algorithm descriptors;
- b) More than one PRF IKEv2-SCSI cryptographic algorithm descriptor;
- c) A PRF IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that does not appear in the SA Creations Capabilities payload (see 7.6.3.5.11) last returned by the device server to any application client; or
- d) An PRF IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that is not shown in table 425.

#### 7.6.3.6.4 Integrity algorithm (INTEG) IKEv2-SCSI cryptographic algorithm descriptor

When the algorithm type field is set to INTEG (i.e., 03h) in an IKEv2-SCSI cryptographic algorithm descriptor (see table 426), the descriptor specifies an integrity checking (i.e., data authentication) algorithm to be applied during the IKEv2-SCSI Authentication step (see 5.14.4.9) and when the SA created by the IKEv2-SCSI is applied to user data.

**Table 426 — INTEG IKEv2-SCSI cryptographic algorithm descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	ALGORITHM TYPE (03h)							
1	Reserved							
2	(MSB)							
3	IKE DESCRIPTOR LENGTH (000Ch)							(LSB)
4	(MSB)							
7	ALGORITHM IDENTIFIER							(LSB)
8	Reserved							
11								

The ALGORITHM TYPE field and IKE DESCRIPTOR LENGTH field are described in 7.6.3.6.1.

The ALGORITHM IDENTIFIER field (see table 427) specifies integrity checking algorithm and shared key length to which the INTEG IKEv2-SCSI cryptographic algorithm descriptor applies.

**Table 427 — INTEG ALGORITHM IDENTIFIER field**

Code	IKEv2 Name	ICV <sup>a</sup> length (bytes)	Key length (bytes)	Support	Reference
8003 0002h	AUTH_HMAC_SHA1_96	12	20	Optional	RFC 2404
8003 000Ch	AUTH_HMAC_SHA2_256_128	16	32	Optional	RFC 4868
8003 000Eh	AUTH_HMAC_SHA2_512_256	32	64	Optional	RFC 4868
F003 0001h	AUTH_COMBINED	n/a	0	Optional	this subclause
8003 0400h to 8003 FFFFh	Vendor Specific				
0000 0000h to 0000 FFFFh	Restricted				IANA
All others	Reserved				

<sup>a</sup> Integrity Check Value (see 3.1.72).

The AUTH\_COMBINED integrity checking algorithm is used with encryption algorithms that include integrity checking as described in 7.6.3.6.2. The AUTH\_COMBINED algorithm identifier specifies that no additional integrity check is performed, as indicated by the zero-length key.

The key length used with an integrity checking algorithm is determined by the algorithm identifier as shown in table 427.

The IKEv2-SCSI CCS state maintained for the I\_T\_L nexus shall be abandoned as described in 7.6.3.8.3 if the parameter list contains:

- a) No INTEG IKEv2-SCSI cryptographic algorithm descriptors;
- b) More than one INTEG IKEv2-SCSI cryptographic algorithm descriptor;
- c) An INTEG IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that does not appear in the SA Creations Capabilities payload (see 7.6.3.5.11) last returned by the device server to any application client; or
- d) An INTEG IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that is not shown in table 427.

7.6.3.6.5 Diffie-Hellman group (D-H) IKEv2-SCSI cryptographic algorithm descriptor

When the algorithm type field is set to D-H (i.e., 04h) in an IKEv2-SCSI cryptographic algorithm descriptor (see table 428), the descriptor specifies Diffie-Hellman group and Diffie-Hellman algorithm used during the IKEv2-SCSI Key Exchange step (see 5.14.4.8) to derive a shared key (see 3.1.155) that is known only to the application client and device server.

Table 428 — D-H IKEv2-SCSI cryptographic algorithm descriptor format

Bit Byte	7	6	5	4	3	2	1	0						
0	ALGORITHM TYPE (04h)													
1	Reserved													
2	(MSB)	IKE DESCRIPTOR LENGTH (000Ch)												
3								(LSB)						
4	(MSB)	ALGORITHM IDENTIFIER												
7								(LSB)						
8	Reserved													
11														

The ALGORITHM TYPE field and IKE DESCRIPTOR LENGTH field are described in 7.6.3.6.1.

The ALGORITHM IDENTIFIER field (see table 429) specifies Diffie-Hellman algorithm, group, and shared key length to which the D-H IKEv2-SCSI cryptographic algorithm descriptor applies.

**Table 429 — D-H ALGORITHM IDENTIFIER field**

Code	Description	Key length (bytes)	Support	Reference
8004 000Eh	2 048-bit MODP group (finite field D-H)	256	Optional	RFC 3526
8004 000Fh	3 072-bit MODP group (finite field D-H)	384	Optional	RFC 3526
8004 0010h	4 096-bit MODP group (finite field D-H)	512	Optional	RFC 3526
8004 0013h	256-bit random ECP group	64	Optional	RFC 4753
8004 0015h	521-bit random ECP group	132	Optional	RFC 4753
8004 0400h to 8004 FFFFh	Vendor Specific			
0000 0000h to 0000 FFFFh	Restricted			IANA
All others	Reserved			

The key length of the public value transferred in the KEY EXCHANGE DATA field (see 7.6.3.5.3) is determined by the algorithm identifier as shown in table 429.

The IKEv2-SCSI CCS state maintained for the I\_T\_L nexus shall be abandoned as described in 7.6.3.8.3 if the parameter list contains:

- a) No D-H IKEv2-SCSI cryptographic algorithm descriptors;
- b) More than one D-H IKEv2-SCSI cryptographic algorithm descriptor;
- c) A D-H IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that does not appear in the SA Creations Capabilities payload (see 7.6.3.5.11) last returned by the device server to any application client; or
- d) An D-H IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that is not shown in table 429.

### 7.6.3.6.6 IKEv2-SCSI authentication algorithm IKEv2-SCSI cryptographic algorithm descriptor

When the algorithm type field is set to SA\_AUTH\_OUT (i.e., F9h) in an IKEv2-SCSI cryptographic algorithm descriptor (see table 430), the descriptor specifies Authentication payload authentication algorithm used by the IKEv2-SCSI Authentication step SECURITY PROTOCOL OUT command (see 5.14.4.9.2).

When the algorithm type field is set to SA\_AUTH\_IN (i.e., FAh) in an IKEv2-SCSI cryptographic algorithm descriptor (see table 430), the descriptor specifies Authentication payload authentication algorithm used by the IKEv2-SCSI Authentication step SECURITY PROTOCOL IN command (see 5.14.4.9.3).

**Table 430 — SA\_AUTH\_OUT and SA\_AUTH\_IN IKEv2-SCSI cryptographic algorithm descriptor format**

Bit Byte	7	6	5	4	3	2	1	0						
0	ALGORITHM TYPE (F9h or FAh)													
1	Reserved													
2	(MSB)	IKE DESCRIPTOR LENGTH (000Ch)												
3								(LSB)						
4	(MSB)	ALGORITHM IDENTIFIER												
7								(LSB)						
8	Reserved													
11														

The ALGORITHM TYPE field and IKE DESCRIPTOR LENGTH field are described in 7.6.3.6.1.

The ALGORITHM IDENTIFIER field (see table 431) specifies Authentication payload authentication algorithm to which the SA\_AUTH\_OUT IKEv2-SCSI cryptographic algorithm descriptor or SA\_AUTH\_IN IKEv2-SCSI cryptographic algorithm descriptor applies.

**Table 431 — SA\_AUTH\_OUT and SA\_AUTH\_IN ALGORITHM IDENTIFIER field**

Code	Description	Support	Reference
00F9 0000h	SA_AUTH_NONE	Optional	this subclause
00F9 0001h	RSA Digital Signature with SHA-1 <sup>a</sup>	Optional	RFC 4306
00F9 0002h	Shared Key Message Integrity Code	Optional	RFC 4306 <sup>b, c</sup>
00F9 0009h	ECDSA with SHA-256 on the P-256 curve <sup>a</sup>	Optional	RFC 4754
00F9 000Bh	ECDSA with SHA-512 on the P-521 curve <sup>a</sup>	Optional	RFC 4754
00F9 00C9h to 00F9 00FFh	Vendor Specific	Optional	
0000 0000h to 0000 FFFFh	Restricted	Prohibited	IANA
All others	Reserved		
<sup>a</sup> Use of certificates with this digital signature authentication algorithm is optional. <sup>b</sup> The 17 ASCII character non-terminated pre-shared key (see 3.1.107) pad string "Key Pad for IKEv2" specified by RFC 4306 is replaced by the 22 ASCII character non-terminated pre-shared key pad string "Key Pad for IKEv2-SCSI". <sup>c</sup> The pre-shared key (see 3.1.107) requirements used by this standard (see 5.14.4.5) apply in addition to those found in RFC 4306.			

SA\_AUTH\_NONE specifies the omission of the IKEv2-SCSI Authentication step (see 5.14.4.9) as follows:

- a) The presence of an SA\_AUTH\_OUT IKEv2-SCSI cryptographic algorithm descriptor and an SA\_AUTH\_IN IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA\_AUTH\_NONE is an SA Creation Capabilities payload (see 7.6.3.5.11) indicates that the device server is allowed to negotiate the omission of the IKEv2-SCSI Authentication step; and
- b) The presence of an SA\_AUTH\_OUT IKEv2-SCSI cryptographic algorithm descriptor and an SA\_AUTH\_IN IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA\_AUTH\_NONE is an IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.6.3.5.12) indicates the following based upon the command whose parameter data carries the payload:
  - A) In the parameter list for a Key Exchange SECURITY PROTOCOL OUT command (see 5.14.4.8.2), SA\_AUTH\_NONE specifies that the application client is requesting that the IKEv2-SCSI Authentication step be skipped; and
  - B) In the parameter data for a Key Exchange SECURITY PROTOCOL IN command (see 5.14.4.8.3), SA\_AUTH\_NONE indicates that the device server has agreed to skip the IKEv2-SCSI Authentication step.

If it is reported by a device server in its capabilities and selected by an application client, then the IKEv2-SCSI Authentication step is skipped and the resulting SAs are not authenticated.

An SA\_AUTH\_OUT IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA\_AUTH\_NONE shall not appear in an IKEv2-SCSI SA Cryptographic Algorithms payload except as described in 5.14.4.6.

The Shared Key Message Integrity Code is based on a pre-shared key (see 3.1.107 and 5.14.4.5) that is associated with the identity in the Identification payload (see 7.6.3.5.4).

The IKEv2-SCSI CCS state maintained for the I\_T\_L nexus shall be abandoned as described in 7.6.3.8.3 if the parameter list contains:

- a) No SA\_AUTH\_OUT IKEv2-SCSI cryptographic algorithm descriptors;
- b) No SA\_AUTH\_IN IKEv2-SCSI cryptographic algorithm descriptors;
- c) More than one SA\_AUTH\_OUT IKEv2-SCSI cryptographic algorithm descriptor;
- d) More than one SA\_AUTH\_IN IKEv2-SCSI cryptographic algorithm descriptor;
- e) An SA\_AUTH\_OUT IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that does not appear in the SA Creations Capabilities payload (see 7.6.3.5.11) last returned by the device server to any application client;
- f) An SA\_AUTH\_IN IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that does not appear in the SA Creations Capabilities payload last returned by the device server to any application client
- g) An SA\_AUTH\_OUT IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that is not shown in table 431;
- h) An SA\_AUTH\_IN IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that is not shown in table 431;
- i) An SA\_AUTH\_OUT IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA\_AUTH\_NONE, and an SA\_AUTH\_IN IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to a value other than SA\_AUTH\_NONE; or
- j) An SA\_AUTH\_IN IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA\_AUTH\_NONE, and an SA\_AUTH\_OUT IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to a value other than SA\_AUTH\_NONE.



### 7.6.3.7 Errors in IKEv2-SCSI security protocol commands

For a single I\_T\_L nexus, the device server shall ensure that the two or four IKEv2-SCSI CCS commands are processed in the order described in 5.14.4.1 based only on the contents of the CDB (i.e., the SECURITY PROTOCOL OUT parameter data shall not be processed unless the tests in table 432 specify the processing of the command) using the tests and responses shown in table 432.

**Table 432 — IKEv2-SCSI command ordering processing requirements on a single I\_T\_L nexus (part 1 of 2)**

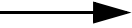

	Time 				
	Before Key Exchange step SECURITY PROTOCOL OUT command returns GOOD status	After a Key Exchange step SECURITY PROTOCOL OUT command returns GOOD status	After a Key Exchange step SECURITY PROTOCOL IN command returns GOOD status	After an Authentication step SECURITY PROTOCOL OUT command returns GOOD status	After an Authentication step SECURITY PROTOCOL IN command returns GOOD status
IKEv2-SCSI SECURITY PROTOCOL OUT command or SECURITY PROTOCOL IN command received					
Key Exchange step SECURITY PROTOCOL OUT command	Process the command as described in this standard	Do not process the command <sup>a</sup>	Do not process the command <sup>a</sup>	Do not process the command <sup>a</sup>	Same as before Key Exchange step SECURITY PROTOCOL OUT command
Key Exchange step SECURITY PROTOCOL IN command	No IKEv2-SCSI CCS exists <sup>b</sup>	Process the command as described in this standard	Repeat processing of the command <sup>c</sup>	Do not process the command <sup>a</sup>	Do not process the command <sup>a</sup>
<sup>a</sup> The command shall be terminated and the IKEv2-SCSI CCS shall be continued as follows: a) The command shall be terminated with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS; and b) The device server shall continue the IKEv2-SCSI CCS by preparing to receive another IKEv2-SCSI CCS SECURITY PROTOCOL OUT command. <sup>b</sup> The command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB. <sup>c</sup> Processing of the SECURITY PROTOCOL IN commands in an IKEv2-SCSI CCS may be repeated. The device server should save the information necessary to repeat processing of these commands until the number of seconds specified in the IKEV2-SCSI PROTOCOL TIMEOUT field of the IKEv2-SCSI Timeout Values payload (see 7.6.3.5.14) have elapsed since the processing of the Authentication step SECURITY PROTOCOL OUT command that completed with GOOD status.					

Table 432 — IKEv2-SCSI command ordering processing requirements on a single I\_T\_L nexus (part 2 of 2)

IKEv2-SCSI SECURITY PROTOCOL OUT command or SECURITY PROTOCOL IN command received	Time 				
	Before Key Exchange step SECURITY PROTOCOL OUT command returns GOOD status	After a Key Exchange step SECURITY PROTOCOL OUT command returns GOOD status	After a Key Exchange step SECURITY PROTOCOL IN command returns GOOD status	After an Authentication step SECURITY PROTOCOL OUT command returns GOOD status	After an Authentication step SECURITY PROTOCOL IN command returns GOOD status
Authentication step SECURITY PROTOCOL OUT command	No IKEv2-SCSI CCS exists <sup>b</sup>	Do not process the command <sup>a</sup>	Process the command as described in this standard	Do not process the command <sup>a</sup>	Do not process the command <sup>a</sup>
Authentication step SECURITY PROTOCOL IN command		Do not process the command <sup>a</sup>	Do not process the command <sup>a</sup>	Process the command as described in this standard	Repeat processing of the command <sup>c</sup>
Command with an invalid field in the CDB	No IKEv2-SCSI CCS exists <sup>b</sup>	Do not process the command <sup>a</sup>	Do not process the command <sup>a</sup>	Do not process the command <sup>a</sup>	No IKEv2-SCSI CCS exists <sup>b</sup>
<sup>a</sup> The command shall be terminated and the IKEv2-SCSI CCS shall be continued as follows: a) The command shall be terminated with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS; and b) The device server shall continue the IKEv2-SCSI CCS by preparing to receive another IKEv2-SCSI CCS SECURITY PROTOCOL OUT command. <sup>b</sup> The command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB. <sup>c</sup> Processing of the SECURITY PROTOCOL IN commands in an IKEv2-SCSI CCS may be repeated. The device server should save the information necessary to repeat processing of these commands until the number of seconds specified in the IKEV2-SCSI PROTOCOL TIMEOUT field of the IKEv2-SCSI Timeout Values payload (see 7.6.3.5.14) have elapsed since the processing of the Authentication step SECURITY PROTOCOL OUT command that completed with GOOD status.					

The processing shown in table 432 shall be performed before parameter data error handling described in 7.6.3.8.

### 7.6.3.8 Errors in IKEv2-SCSI security protocol parameter data

#### 7.6.3.8.1 Overview

Errors in the parameter data transferred to the device server by an IKEv2-SCSI SECURITY PROTOCOL OUT command are classified (see table 433) based on the ease with which they may be used to mount denial of service attacks against IKEv2-SCSI SA creation operations by an attacker that has not participated as the application client or device server in the Key Exchange step SECURITY PROTOCOL OUT command (see 5.14.4.8.2) that started the IKEv2-SCSI CCS.

**Table 433 — IKEv2-SCSI parameter error categories**

Denial of service attack potential	Applicable SECURITY PROTOCOL OUT commands	Error handling description	Reference
High <sup>a</sup>	Authentication step, and Delete operation	If possible, the IKEv2-SCSI CCS state maintained for an I_T_L nexus is not changed	7.6.3.8.2
Minimal <sup>b</sup>	Key Exchange step, Authentication step, and Delete operation	The IKEv2-SCSI CCS state maintained for an I_T_L nexus is abandoned	7.6.3.8.3
<sup>a</sup> Attacks capable of causing significant harm by sending a malformed IKEv2-SCSI SECURITY PROTOCOL OUT command to the device server. <sup>b</sup> Attacks that produce no significant harm, or collusive attacks (i.e., attacks that require knowledge of the IKEv2-SCSI CCS shared keys and participation in the IKEv2-SCSI CCS). Collusive attacks depend on collusion between the attacker and the application client (i.e., require the application client to act against its own best interests). Abandoning the IKEv2-SCSI CCS is a justified response to such attacks.			

#### 7.6.3.8.2 Errors with high denial of service attack potential

Errors detected before or during the decryption and integrity checking of an Encrypted payload in an Authentication step SECURITY PROTOCOL OUT command or a Delete operation SECURITY PROTOCOL OUT command have a high potential for being a denial of service attack against one or more application clients (see table 433 in 7.6.3.8.1).

The device server shall respond to these SECURITY PROTOCOL OUT command errors as follows:

- The command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to SA CREATION PARAMETER VALUE REJECTED; and
- The device server shall continue the IKEv2-SCSI CCS, if any, by preparing to receive an Authentication step SECURITY PROTOCOL OUT command.

If a specific field is the cause for returning the SA CREATION PARAMETER VALUE REJECTED additional sense code, then the SKSV bit may be set to one and SENSE KEY SPECIFIC field shall be set as defined in 4.5.2.4.2.

#### 7.6.3.8.3 Errors with minimal denial of service attack potential

The errors that have minimal denial of service attack potential for IKEv2-SCSI SA creation (see table 433 in 7.6.3.8.1) are:

- a) All errors detected for a Key Exchange step SECURITY PROTOCOL OUT command or its associated parameter data (e.g., errors detected in the IKEv2-SCSI header) that is attempting to establish a new IKEv2-SCSI CCS on an I\_T\_L nexus; and
- b) All errors that are detected after the Encrypted payload has been successfully decrypted and integrity checked in an Authentication step SECURITY PROTOCOL OUT command (see 5.14.4.9.2) or a Delete operation SECURITY PROTOCOL OUT command (see 5.14.4.13).

The device server shall respond to these SECURITY PROTOCOL OUT command errors by terminating the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to SA CREATION PARAMETER VALUE INVALID. The state being maintained for an IKEv2-SCSI CCS on the I\_T\_L nexus on which the command was received, if any, shall be abandoned (see 5.14.4.12).

If a specific field is the cause for returning the SA CREATION PARAMETER VALUE INVALID additional sense code, then the SKSV bit shall be set to one and SENSE KEY SPECIFIC field shall be set as defined in 4.5.2.4.2.

### 7.6.3.9 Translating IKEv2 errors

IKEv2 (see RFC 4306) defines an error reporting mechanism based on the Notify payload. This standard translates such error reports into the device server and application actions defined in this subclause.

If a device server is required by IKEv2 to report an error using a Notify payload, the device server shall translate error into CHECK CONDITION status with the sense key and additional sense code shown in table 434. The device server shall terminate the SECURITY PROTOCOL OUT command (see 6.31) that transferred the parameter list in which the IKE requirements for one or more payloads (see 7.6.3.5) require use of the Notify payload to report an error. The SECURITY PROTOCOL OUT command shall be terminated as described in this subclause. The device server shall not report the IKE errors described in this subclause by terminating a SECURITY PROTOCOL IN command (see 6.30) with CHECK CONDITION status.

**Table 434 — IKEv2 Notify payload error translations for IKEv2-SCSI**

IKEv2 (see RFC 4306)		IKEv2-SCSI	
Error Type	Description	Additional sense code	Sense key
0000h	Reserved		
0001h	UNSUPPORTED_CRITICAL_PAYLOAD	SA CREATION PARAMETER NOT SUPPORTED	ILLEGAL REQUEST
0004h	INVALID_IKE_SPI	SA CREATION PARAMETER VALUE INVALID or SA CREATION PARAMETER VALUE REJECTED	
0005h	INVALID_MAJOR_VERSION		
0007h	INVALID_SYNTAX <sup>a</sup>		
0009h	INVALID_MESSAGE_ID	SA CREATION PARAMETER VALUE REJECTED	
000Bh	INVALID_SPI	SA CREATION PARAMETER VALUE INVALID <sup>b</sup>	
000Eh	NO_PROPOSAL_CHOSEN <sup>c</sup>	SA CREATION PARAMETER VALUE INVALID	
0011h	INVALID_KE_PAYLOAD <sup>c</sup>		
0018h	AUTHENTICATION_FAILED	AUTHENTICATION FAILED	ABORTED COMMAND
0022h to 0027h	See RFC 4306 <sup>d</sup>	n/a	n/a
2000h to 3FFFh	Vendor Specific		
All others	Restricted		

<sup>a</sup> This sense key and one of the additional sense codes shown shall be returned for a syntax error within an Encrypted payload (see 7.6.3.5.10) regardless of conflicting IKEv2 requirements.

<sup>b</sup> SA CREATION PARAMETER VALUE INVALID shall be used for an invalid SAI in an IKEv2-SCSI SECURITY PROTOCOL IN or SECURITY PROTOCOL OUT. The additional sense code for an invalid SAI in all other commands is specified by the applicable usage type definition (see table 58 in 5.14.2.2).

<sup>c</sup> An application client recovers by restarting processing with the Device Capabilities step (see 5.14.4.7) to rediscover the device server's capabilities.

<sup>d</sup> These IKEv2 Error Types are used for features that are not supported by IKEv2-SCSI SA creation.

If an application client detects an IKEv2 error that RFC 4306 requires to be reported with a Notify payload, the application client then the application client should abandon the IKEv2-SCSI CCS and notify the device server that it is abandoning the IKEv2-SCSI CCS as described in 5.14.4.12.

## 7.6.4 CbCS security protocol

### 7.6.4.1 Overview

If the SECURITY PROTOCOL field in a SECURITY PROTOCOL IN command (see 6.30) is set to 07h, then the command specifies one of the CbCS pages (see 7.6.4.2) to be returned by the device sever. The information returned by a CbCS SECURITY PROTOCOL IN command indicates the CbCS operating parameters of:

- a) The logical unit to which the CbCS SECURITY PROTOCOL IN command is addressed; or
- b) The SCSI target device that contains the well-known logical unit to which the CbCS SECURITY PROTOCOL IN command is addressed.

If the SECURITY PROTOCOL field in a SECURITY PROTOCOL OUT command (see 6.31) is set to 07h, then the command specifies one of the CbCS pages (see 7.6.4.4) to be sent to the device sever. The instructions sent in a CbCS SECURITY PROTOCOL OUT command specify the CbCS operating parameters of:

- a) The logical unit to which the CbCS SECURITY PROTOCOL OUT command is addressed; or
- b) The SCSI target device that contains the well-known logical unit to which the CbCS SECURITY PROTOCOL OUT command is addressed.

### 7.6.4.2 CbCS SECURITY PROTOCOL IN CDB description

The CbCS SECURITY PROTOCOL IN CDB has the format defined in 6.30 with the additional requirements described in this subclause.

When the SECURITY PROTOCOL field is set to CbCS (i.e., 07h) in a SECURITY PROTOCOL IN command, the SECURITY PROTOCOL SPECIFIC field (see table 435) specifies the CbCS page to be returned in the parameter data (see 7.6.4.3). If the CBCS bit is set to one in the Extended INQUIRY Data VPD page (see 7.7.4), the CbCS SECURITY PROTOCOL IN command support requirements are shown in table 435.

**Table 435 — SECURITY PROTOCOL SPECIFIC field for the CbCS SECURITY PROTOCOL IN command**

Code <sup>a</sup>	CbCS page returned	Support	Reference
0000h	Supported CbCS SECURITY PROTOCOL IN Pages	Mandatory	7.6.4.3.1
0001h	Supported CbCS SECURITY PROTOCOL OUT Pages	Mandatory	7.6.4.3.2
0002h	Unchangeable CbCS Parameters	Mandatory	7.6.4.3.3
0003h to 003Eh	Reserved		
3Fh	Security Token	Optional <sup>b</sup>	7.6.4.3.4
0040h	Current CbCS Parameters	Mandatory	7.6.4.3.5
0041h to D00Fh	Reserved		
D010h	Set Master Key – Seed Exchange	Optional <sup>b</sup>	7.6.4.3.6
D011h to FFFFh	Reserved		

<sup>a</sup> If the SECURITY PROTOCOL SPECIFIC field contains a value that is less than D000h, then the working key specified by the KEY VERSION field in the CbCS capability descriptor (see 6.19.2.3) shall be used to compute the capability key (see 5.14.6.8.12). If the SECURITY PROTOCOL SPECIFIC field contains a value that is greater than or equal to D000h, then the authentication key component of the master key (see 5.14.6.8.11) shall be used to compute the capability key.

<sup>b</sup> Mandatory if the CAPKEY CbCS method (see 5.14.6.8.8.3) is supported.

If a CbCS SECURITY PROTOCOL IN command is received with the INC\_512 bit set to one, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

### 7.6.4.3 CbCS SECURITY PROTOCOL IN parameter data

#### 7.6.4.3.1 Supported CbCS SECURITY PROTOCOL IN Pages CbCS page

The Supported CbCS SECURITY PROTOCOL IN Pages CbCS page (see table 436) lists the CbCS pages that are supported for the (i.e., the values that are allowed in the SECURITY PROTOCOL SPECIFIC field in a) SECURITY PROTOCOL IN command (see 6.30).

**Table 436 — Supported CbCS SECURITY PROTOCOL IN Pages CbCS page format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)	PAGE CODE (0000h)						(LSB)
1								
2	(MSB)	PAGE LENGTH (n-3)						(LSB)
3								
Supported CbCS SECURITY PROTOCOL IN page list								
4	(MSB)	SUPPORTED CBCS SECURITY PROTOCOL IN PAGE [first]						(LSB)
5		(0000h)						
		⋮						
n-1	(MSB)	SUPPORTED CBCS SECURITY PROTOCOL IN PAGE [last]						(LSB)
n								

The PAGE CODE field shall be set to 0000h to indicate that the Supported CbCS SECURITY PROTOCOL IN Pages CbCS page is being returned.

The page length field indicates the number of bytes that follow in the Supported CbCS SECURITY PROTOCOL IN Pages CbCS page.

Each SUPPORTED CBCS SECURITY PROTOCOL IN PAGE field indicates the page code (see table 435 in 7.6.4.2) of one CbCS page that is supported by the SECURITY PROTOCOL IN command when the SECURITY PROTOCOL field (see 6.30) is set to 07h (i.e., CbCS). The values in the SUPPORTED CBCS SECURITY PROTOCOL IN PAGE fields shall be returned in ascending order beginning with 0000h (i.e., this CbCS page).

#### 7.6.4.3.2 Supported CbCS SECURITY PROTOCOL OUT pages CbCS page

The Supported CbCS SECURITY PROTOCOL OUT Pages CbCS page (see table 437) lists the CbCS pages that are supported for the (i.e., the values that are allowed in the SECURITY PROTOCOL SPECIFIC field in a) SECURITY PROTOCOL OUT command (see 6.31).

**Table 437 — Supported CbCS SECURITY PROTOCOL OUT Pages CbCS page format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)	PAGE CODE (0001h)						(LSB)
1								
2	(MSB)	PAGE LENGTH (n-3)						(LSB)
3								
Supported CbCS SECURITY PROTOCOL OUT page list								
4	(MSB)	SUPPORTED CBCS SECURITY PROTOCOL OUT PAGE						(LSB)
5		[first]						
		⋮						
n-1	(MSB)	SUPPORTED CBCS SECURITY PROTOCOL OUT PAGE						(LSB)
n		[last]						

The PAGE CODE field shall be set to 0001h to indicate that the Supported CbCS SECURITY PROTOCOL OUT Pages CbCS page is being returned.

The page length field indicates the number of bytes that follow in the Supported CbCS SECURITY PROTOCOL OUT Pages CbCS page.

Each SUPPORTED CBCS SECURITY PROTOCOL OUT PAGE field indicates the page code (see table 444 in 7.6.4.4) of one CbCS page that is supported by the SECURITY PROTOCOL OUT command when the SECURITY PROTOCOL field (see 6.31) is set to 07h (i.e., CbCS). The values in the SUPPORTED CBCS SECURITY PROTOCOL IN PAGE fields shall be returned in ascending order.



### 7.6.4.3.3 Unchangeable CbCS Parameters CbCS page

The Unchangeable CbCS Parameters CbCS page (see table 438) indicates the supported CbCS features and algorithms.

**Table 438 — Unchangeable CbCS Parameters CbCS page format (part 1 of 2)**

Bit Byte	7	6	5	4	3	2	1	0	
0	(MSB)	PAGE CODE (0002h)							(LSB)
1									
2	(MSB)	PAGE LENGTH (n-3)							(LSB)
3									
4	KEYS SUPPORT		MIN CBCS METHOD SUP		Reserved				
5	Reserved								
6	(MSB)	SUPPORTED INTEGRITY CHECK VALUE ALGORITHM							(LSB)
7		LIST LENGTH (c-7)							(LSB)
Supported integrity check value algorithms list									
8	(MSB)	SUPPORTED INTEGRITY CHECK VALUE ALGORITHM							(LSB)
11		[first]							
		:							
		:							
c-3	(MSB)	SUPPORTED INTEGRITY CHECK VALUE ALGORITHM							(LSB)
c		[last]							
c+1		Reserved							
c+2									
c+3	(MSB)	SUPPORTED D-H ALGORITHM LIST LENGTH (d-c-4)							(LSB)
c+4									
Supported Diffie-Hellman (D-H) algorithms list									
c+5	(MSB)	SUPPORTED D-H ALGORITHM [first]							(LSB)
c+8									
		:							
		:							
d-3	(MSB)	SUPPORTED D-H ALGORITHM [last]							(LSB)
d									
d+1	(MSB)	SUPPORTED CBCS METHODS LIST LENGTH (n-d-2)							(LSB)
d+2									

**Table 438 — Unchangeable CbCS Parameters CbCS page format (part 2 of 2)**

Bit Byte	7	6	5	4	3	2	1	0
Supported CbCS methods list								
d+3	SUPPORTED CBCS METHOD [first]							
	⋮							
n	SUPPORTED CBCS METHOD [last]							

The PAGE CODE field shall be set to 0002h to indicate that the Unchangeable CbCS Parameters CbCS page is being returned.

The page length field indicates the number of bytes that follow in the Unchangeable CbCS Parameters CbCS page.

The KEYS SUPPORT field (see table 439) indicates the type of CbCS master keys and working keys supported.

**Table 439 — KEYS SUPPORT field**

Code	Description
00b	Reserved
01b	The SCSI target device supports single CbCS master key and a set of CbCS working keys (see 5.14.6.8.11) for the SCSI target device, but the logical units in the SCSI target device do not support CbCS master keys or working keys.
10b	The SCSI target device does not support CbCS master keys or working keys for the SCSI target device, but each logical unit in the SCSI target device supports a single CbCS master key and a set of CbCS working keys for that logical unit.
11b	The SCSI target device supports single CbCS master key and a set of CbCS working keys for the SCSI target device, and each logical unit in the SCSI target device supports a single CbCS master key and a set of CbCS working keys for that logical unit. Keys stored in the logical unit take precedence over keys stored in the SCSI target device (see 5.14.6.8.6).

The MIN CBCS METHOD SUP field (see table 440) indicates how the assignment of the minimum allowable CbCS method is supported.

**Table 440 — MIN CBCS METHOD SUP field**

Code	Description
00b	Reserved
01b	A single minimum allowed CbCS method (see 5.14.6.8.8) is assigned to all logical units in the SCSI target device, and the SECURITY PROTOCOL well known logical unit is implemented to control its value.
10b	Each logical unit in the SCSI target device is assigned its own minimum allowed CbCS method.
11b	Reserved

The SUPPORTED INTEGRITY CHECK VALUE ALGORITHM LIST LENGTH field indicates the number of bytes that follow in the supported integrity check value algorithms list.

Each SUPPORTED INTEGRITY CHECK VALUE ALGORITHM field indicates one supported algorithm for computing CbCS integrity check values. The values in the SUPPORTED INTEGRITY CHECK VALUE ALGORITHM fields are selected from the codes that table 88 (see 5.14.8) lists as integrity checking (i.e., AUTH) algorithms, except for AUTH\_COMBINED.

The SUPPORTED D-H ALGORITHM LIST LENGTH field indicates the number of bytes that follow in the supported Diffie-Hellman (D-H) algorithms list.

Each SUPPORTED D-H ALGORITHM field indicates one supported algorithm for constructing CbCS shared keys. The values in the SUPPORTED D-H ALGORITHM fields are selected from the codes that table 88 (see 5.14.8) lists as Diffie-Hellman algorithms with finite field D-H computations.

The SUPPORTED CBCS METHODS LIST LENGTH field indicates the number of bytes that follow in the supported CbCS methods list.

Each SUPPORTED CBCS METHODS field indicates one supported CbCS method (see 6.19.2.3). The values in the SUPPORTED CBCS METHODS fields are selected from the codes listed in table 212 (see 6.19.2.3).

#### 7.6.4.3.4 Security Token CbCS page

The Security Token CbCS page (see table 441) indicates the value of the security token (see 5.14.6.8.10) for the I\_T nexus on which the SECURITY PROTOCOL IN command was received.

**Table 441 — Security Token CbCS page format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)	PAGE CODE (003Fh)						(LSB)
1		PAGE LENGTH (n-3)						(LSB)
2	(MSB)	SECURITY TOKEN						(LSB)
3								
4	(MSB)							
n								(LSB)

The PAGE CODE field shall be set to 003Fh to indicate that the Security Token CbCS page is being returned.

The page length field indicates the number of bytes that follow in the Security Token CbCS page.

The SECURITY TOKEN field shall contain the security token (see 5.14.6.8.10) for the I\_T nexus on which the command was received.

#### 7.6.4.3.5 Current CbCS Parameters CbCS page

The Current CbCS Parameters CbCS page (see table 442) indicates the current values for the CbCS parameters (see 5.14.6.8.15) used by the SCSI target device or logical unit as follows:

- a) If the logical unit to which the SECURITY PROTOCOL IN command is addressed is the SECURITY PROTOCOL well known logical unit (see 8.5), then the contents of the Current CbCS Parameters CbCS page apply to the SCSI target device; or

- b) If the logical unit to which the SECURITY PROTOCOL IN command is addressed is not the SECURITY PROTOCOL well known logical unit, then the contents of the Current CbCS Parameters CbCS page apply to the addressed logical unit.

**Table 442 — Current CbCS Parameters CbCS page format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)	PAGE CODE (0040h)						(LSB)
1								
2	(MSB)	PAGE LENGTH (009Ah)						(LSB)
3								
4		Reserved						
6								
7		MINIMUM ALLOWED CBCS METHOD						
8	(MSB)							
11		POLICY ACCESS TAG						(LSB)
12								
15		Reserved						
16	(MSB)							
23		MASTER KEY IDENTIFIER						(LSB)
24	(MSB)							
31		WORKING KEY 0 IDENTIFIER						(LSB)
32	(MSB)							
39		WORKING KEY 1 IDENTIFIER						(LSB)
		⋮						
144	(MSB)							
151		WORKING KEY 15 IDENTIFIER						(LSB)
152	(MSB)							
157		CLOCK						(LSB)

The PAGE CODE field shall be set to 0040h to indicate that the Current CbCS Parameters CbCS page is being returned.

The page length field indicates the number of bytes that follow in the Current CbCS Parameters CbCS page.

The contents of the MINIMUM ALLOWED CBCS METHOD field depend on the logical unit that returned the Current CbCS Parameters CbCS page as follows:

- a) If the logical unit is not the SECURITY PROTOCOL well known logical unit, then the MINIMUM ALLOWED CBCS METHOD field indicates the smallest value allowed in the CBCS METHOD field (see 6.19.2.3) of a capability descriptor processed by the enforcement manager (see 5.14.6.8.7) as described in 5.14.6.8.13.2 (i.e., the value of the minimum CbCS method CbCS parameter for the logical unit (see 5.14.6.8.15); or

- b) If the SECURITY PROTOCOL well known logical unit is returning the Current CbCS Parameters CbCS page, then the MINIMUM ALLOWED CBCS METHOD field indicates the value that will be copied to the minimum CbCS method CbCS parameter of any dynamically created logical units (i.e., the initial minimum CbCS method CbCS parameter summarized in 5.14.6.8.15).

The value in the MINIMUM ALLOWED CBCS METHOD field is selected from those listed in table 212 (see 6.19.2.3).

The contents of the POLICY ACCESS TAG field depend on the logical unit that returned the Current CbCS Parameters CbCS page as follows:

- a) If the logical unit is not the SECURITY PROTOCOL well known logical unit, then the POLICY ACCESS TAG field indicates the value required in the POLICY ACCESS TAG field (see 6.19.2.3) of a capability descriptor processed by the enforcement manager (see 5.14.6.8.7) as described in 5.14.6.8.13.2 (i.e., the value of the policy access tag CbCS parameter for the logical unit (see 5.14.6.8.15); or
- b) If the SECURITY PROTOCOL well known logical unit is returning the Current CbCS Parameters CbCS page, then the POLICY ACCESS TAG field indicates the value that will be copied to the policy access tag CbCS parameter of any dynamically created logical units (i.e., the initial policy access tag CbCS parameter summarized in 5.14.6.8.15).

The MASTER KEY IDENTIFIER field contains the current CbCS shared key identifier (see 5.14.6.8.11.2) for the master key (see 5.14.6.8.11).

The WORKING KEY 0 IDENTIFIER field, WORKING KEY 1 IDENTIFIER field, WORKING KEY 2 IDENTIFIER field, ... and WORKING KEY 15 IDENTIFIER field contain the current CbCS shared key identifier (see 5.14.6.8.11.2) for the working keys (see 5.14.6.8.11).

The CLOCK field shall be set to the TIMESTAMP field format and value defined in 5.13.

#### 7.6.4.3.6 Set Master Key – Seed Exchange CbCS page

The Set Master Key – Seed Exchange CbCS page (see table 443) in a SECURITY PROTOCOL IN command continues a CbCS master key update CCS (see 5.14.6.8.11.4) that was initiated by processing of a Set Master Key – Seed Exchange CbCS page in a SECURITY PROTOCOL OUT command (see 7.6.4.5.5), and completes a Diffie-Hellman key exchange protocol as part of that CCS.

**Table 443 — Set Master Key – Seed Exchange CbCS page format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1	PAGE CODE (D010h)							(LSB)
2	(MSB)							
3	PAGE LENGTH (n-3)							(LSB)
4	(MSB)							
n	D-H DATA							(LSB)

The Diffie-Hellman (D-H) algorithm being used in the CbCS master key update CCS is determined by the contents of the D-H ALGORITHM field in the Set Master Key – Seed Exchange CbCS page in the SECURITY PROTOCOL OUT command (see 7.6.4.5.5) that initiated the CCS. This Diffie-Hellman algorithm specifies the DH\_generator value and DH\_prime value to be used in the computation of the D-H DATA field as described in this subclause.

The PAGE CODE field set to D010h specifies that the Set Master Key – Seed Exchange CbCS page follows.

The PAGE LENGTH field specifies the number of bytes that follow in the Set Master Key – Seed Exchange CbCS page. If the PAGE LENGTH field shall be set to a value that is inconsistent with the Diffie-Hellman algorithm specified for the CbCS master key update CCS, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The contents of the D-H DATA field are computed as follows:

- 1) A random number,  $y$ , is generated having a value between 0 and  $DH\_prime$  minus one observing the requirements in RFC 4086; and
- 2) The D-H DATA field is set to  $DH\_generator^y$  modulo  $DH\_prime$ , where the  $DH\_generator$  and  $DH\_prime$  values are identified by the value in the D-H ALGORITHM field.

As part of the successful completion of processing for the SECURITY PROTOCOL IN command transfer of the Set Master Key – Seed Exchange CbCS page the device server and application client store as part of the state maintained for CbCS master key update CCS (see 5.14.6.8.11.4):

- a) The authentication key component of the next master key; and
- b) The generation key component of the next master key.

The new master key is computed as follows:

- 1) An initial\_seed value that is the value  $DH\_generator^x$  modulo  $DH\_prime$  is computed as follows:
  - A) The device server computes  $(DH\_generator^x \text{ modulo } DH\_prime)^y$ , where  $DH\_generator^x$  is the contents of the D-H DATA field in the SECURITY PROTOCOL OUT command and  $y$  is the random number generated by the device server; and
  - B) The application client computes  $(DH\_generator^y \text{ modulo } DH\_prime)^x$ , where  $DH\_generator^y$  is the contents of the D-H DATA field in the SECURITY PROTOCOL IN command and  $x$  is the random number generated by the application client (see 7.6.4.5.5);
- 2) The generation key component of the new master key is computed using the integrity check value algorithm specified by the integrity check value algorithm field in the capability (see 6.19.2.3) in the CbCS extension descriptor (see 5.14.6.8.16) associated with the SECURITY PROTOCOL IN command that transferred the Set Master Key – Seed Exchange CbCS page. The following inputs are used with the specified integrity check value algorithm:
  - A) The concatenation of the initial\_seed value computed in step 1) and all of the bytes in the Device Identification VPD page (see 7.7.3) for the logical unit that is processing the CbCS master key update CCS (see 5.14.6.8.11.4) as the string for which the integrity check value is to be computed; and
  - B) The generation key component of the current master key (see 5.14.6.8.11) for the logical unit that is processing the CbCS master key update CCS as the cryptographic key;
- 3) A modified\_seed value is computed as follows:
  - A) If the least significant bit of the initial\_seed value is zero, then the modified\_seed value is equal to the initial\_seed value with the least significant bit set to one; and
  - B) If the least significant bit of the initial\_seed value is one, then the modified\_seed value is equal to the initial\_seed value with the least significant bit set to zero;
 and
- 4) The authentication key component of the new master key is computed using the integrity check value algorithm specified by the integrity check value algorithm field in the capability (see 6.19.2.3) in the CbCS extension descriptor (see 5.14.6.8.16) associated with the SECURITY PROTOCOL IN command that transferred the Set Master Key – Seed Exchange CbCS page. The following inputs are used with the specified integrity check value algorithm:

- A) The concatenation of the modified\_seed value computed in step 3) and all of the bytes in the Device Identification VPD page for the logical unit that is processing the CbCS master key update CCS as the string for which the integrity check value is to be computed; and
- B) The generation key component of the current master key for the logical unit that is processing the CbCS master key update CCS as the cryptographic key.

#### 7.6.4.4 CbCS SECURITY PROTOCOL OUT CDB description

The CbCS SECURITY PROTOCOL OUT CDB has the format defined in 6.31 with the additional requirements described in this subclause.

When the SECURITY PROTOCOL field is set to CbCS (i.e., 07h) in a SECURITY PROTOCOL OUT command, the SECURITY PROTOCOL SPECIFIC field (see table 444) specifies the CbCS page to be returned in the parameter data (see 7.6.4.5). If the CBCS bit is set to one in the Extended INQUIRY Data VPD page (see 7.7.4), the CbCS SECURITY PROTOCOL IN command support requirements are shown in table 444.

**Table 444 — SECURITY PROTOCOL SPECIFIC field for the CbCS SECURITY PROTOCOL OUT command**

Code <sup>a</sup>	CbCS page sent	Support	Reference
0000h to 0040h	Reserved		
0041h	Set Policy Access Tag	Optional	7.6.4.5.1
0042h	Set Minimum CbCS Method	Optional	7.6.4.5.2
0043h to CFFFh	Reserved		
D000h	Invalidate Key	Optional <sup>b</sup>	7.6.4.5.3
D001h	Set Key	Optional <sup>b</sup>	7.6.4.5.4
D003h to D00Fh	Reserved		
D010h	Set Master Key – Seed Exchange	Optional <sup>b</sup>	7.6.4.5.5
D011h	Set Master Key – Change Master Key	Optional <sup>b</sup>	7.6.4.5.6
D012h to FFFFh	Reserved		
<sup>a</sup> If the SECURITY PROTOCOL SPECIFIC field contains a value that is less than D000h, then the working key specified by the KEY VERSION field in the CbCS capability descriptor (see 6.19.2.3) shall be used to compute the capability key (see 5.14.6.8.12). If the SECURITY PROTOCOL SPECIFIC field contains a value that is greater than or equal to D000h, then the authentication key component of the master key (see 5.14.6.8.11) shall be used to compute the capability key. <sup>b</sup> Mandatory if the CAPKEY CbCS method (see 5.14.6.8.8.3) is supported.			

If a CbCS SECURITY PROTOCOL OUT command is received with the INC\_512 bit set to one, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

#### 7.6.4.5 CbCS SECURITY PROTOCOL OUT parameter data

##### 7.6.4.5.1 Set Policy Access Tag CbCS page

The Set Policy Access Tag CbCS page (see table 445) specifies a new policy access tag CbCS parameter value or a new initial policy access tag CbCS parameter value.

**Table 445 — Set Policy Access Tag CbCS page format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____							
1	PAGE CODE (0041h) _____ (LSB)							
2	(MSB) _____							
3	PAGE LENGTH (0004h) _____ (LSB)							
4	(MSB) _____							
7	POLICY ACCESS TAG _____ (LSB)							

The PAGE CODE field set to 0041h specifies that the Set Policy Access Tag CbCS page follows.

The PAGE LENGTH field specifies the number of bytes that follow in the Set Policy Access Tag CbCS page. If the PAGE LENGTH field is set to a value that is smaller than four, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The CbCS parameter (see 5.14.6.8.15) that is set to the contents of the POLICY ACCESS TAG field depends on the logical unit that is processing the Set Policy Access Tag CbCS page as follows:

- If the logical unit is not the SECURITY PROTOCOL well known logical unit, then the contents of the POLICY ACCESS TAG field shall be placed in the policy access tag CbCS parameter for the logical unit; or
- If the SECURITY PROTOCOL well known logical unit is processing the Set Policy Access Tag CbCS page, then the contents of the POLICY ACCESS TAG field shall be placed in the initial policy access tag CbCS parameter.

##### 7.6.4.5.2 Set Minimum CbCS Method CbCS page

The Set Minimum CbCS Method CbCS page (see table 446) specifies a new minimum CbCS method CbCS parameter value or a new initial minimum CbCS method CbCS parameter value.

**Table 446 — Set Minimum CbCS Method CbCS page format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____							
1	PAGE CODE (0042h) _____ (LSB)							
2	(MSB) _____							
3	PAGE LENGTH (0002h) _____ (LSB)							
4	Reserved							
5	MINIMUM ALLOWED CBCS METHOD							



The PAGE CODE field set to 0042h specifies that the Set Minimum CbCS Method CbCS page follows.

The PAGE LENGTH field specifies the number of bytes that follow in the Set Minimum CbCS Method CbCS page. If the PAGE LENGTH field is set to a value that is smaller than two, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The CbCS parameter or CbCS parameters (see 5.14.6.8.15) that are set to the contents of the MINIMUM ALLOWED CBCS METHOD field depends on the logical unit that is processing the Set Minimum CbCS Method CbCS page and the contents of the MIN CBCS METHOD SUP field in the Unchangeable CbCS Parameters CbCS page (see 7.6.4.3.3) as shown in table 447.

**Table 447 — Minimum CbCS Method CbCS Parameter set**

MIN CBCS METHOD SUP field	CbCS Parameter set based on the logical unit that processes the Set Minimum CbCS Method CbCS page	
	Not the SECURITY PROTOCOL well known logical unit	SECURITY PROTOCOL well known logical unit
00b	Reserved (see table 440)	
01b	The command is terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST	All minimum CbCS method CbCS parameters for all logical units in the SCSI target device, and the initial minimum CbCS method, if any
10b	The minimum CbCS method CbCS parameter for the logical unit that processes the Set Minimum CbCS Method CbCS page	The initial minimum CbCS method CbCS parameter
11b	Reserved (see table 440)	

If the MINIMUM ALLOWED CBCS METHOD field specifies a value that does not appear in the supported CbCS methods list in the Unchangeable CbCS Parameters CbCS page (see 7.6.4.3.3), then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

#### 7.6.4.5.3 Invalidate Key CbCS page

The Invalidate Key CbCS page (see table 448) causes a working key (see 5.14.6.8.11) to be invalidated. After the successful processing of an Invalidate Key CbCS page, the working key with the specified key version shall not be

valid for the purposes of enforcement manager (see 5.14.6.8.7) CbCS extension descriptor validation (see 5.14.6.8.13.2).

**Table 448 — Invalidate Key CbCS page format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____							
1	PAGE CODE (D000h) _____							(LSB)
2	(MSB) _____							
3	PAGE LENGTH (0004h) _____							(LSB)
4	Reserved _____							
6								
7	Reserved				KEY VERSION			

The PAGE CODE field set to D000h specifies that the Invalidate Key CbCS page follows.

The PAGE LENGTH field specifies the number of bytes that follow in the Invalidate Key CbCS page. If the PAGE LENGTH field is set to a value that is smaller than four, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The KEY VERSION field specifies which working key (e.g., a KEY VERSION field set to four specifies that the working key whose Current CbCS Parameters CbCS page (see 7.6.4.3.5) key identifier is returned in the WORKING KEY 4 IDENTIFIER field) is to be invalidated as follows:

- a) If the Invalidate Key CbCS page is processed by a logical unit that is not the SECURITY PROTOCOL well known logical unit, then the specified working key for that logical unit shall be invalidated; or
- b) If the Invalidate Key CbCS page is processed by the SECURITY PROTOCOL well known logical unit, then the specified target-wide working key (see 5.14.6.8.11) shall be invalidated.

The CbCS shared key identifier (see 5.14.6.8.11.2) for the invalidated working key shall be set to FFFF FFFF FFFF FFEh.

It shall not be an error to invalidate a working key that is already invalid.

#### 7.6.4.5.4 Set Key CbCS page

The Set Key CbCS page (see table 449) causes a working key (see 5.14.6.8.11) to be set to a new value. After the successful processing of a Set Key CbCS page, the working key with the specified key version shall be valid for the purposes of enforcement manager (see 5.14.6.8.7) CbCS extension descriptor validation (see 5.14.6.8.13.2).

**Table 449 — Set Key CbCS page format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____							
1	PAGE CODE (D001h) _____							(LSB)
2	(MSB) _____							
3	PAGE LENGTH (0020h) _____							(LSB)
4	_____							
6	Reserved _____							
7	Reserved				KEY VERSION			
8	(MSB) _____							
15	KEY IDENTIFIER _____							(LSB)
16	(MSB) _____							
35	SEED _____							(LSB)

The PAGE CODE field set to D001h specifies that the Set Key CbCS page follows.

The PAGE LENGTH field specifies the number of bytes that follow in the Set Key CbCS page. If the PAGE LENGTH field is set to a value that is other than 32, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The KEY VERSION field specifies which working key (e.g., a KEY VERSION field set to six specifies that the working key whose Current CbCS Parameters CbCS page (see 7.6.4.3.5) key identifier is returned in the WORKING KEY 6 IDENTIFIER field) is to be set as follows:

- If the Set Key CbCS page is processed by a logical unit that is not the SECURITY PROTOCOL well known logical unit, then the specified working key for that logical unit shall be set; or
- If the Set Key CbCS page is processed by the SECURITY PROTOCOL well known logical unit, then the specified target-wide working key (see 5.14.6.8.11) shall be set.

The KEY IDENTIFIER field specifies the value to which the CbCS shared key identifier (see 5.14.6.8.11.2) shall be set for the working key affected by the Set Key CbCS page. If the KEY IDENTIFIER field is set to a value that table 72 (see 5.14.6.8.11.2) describes as reserved in the CbCS pages that change CbCS shared key values, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The seed field contains a random number that is an input to the computation of the new working key value.

The value to which the specified working key is set shall be computed using the integrity check value algorithm specified by the INTEGRITY CHECK VALUE ALGORITHM field in the capability (see 6.19.2.3) in the CbCS extension

descriptor (see 5.14.6.8.16) associated with the SECURITY PROTOCOL OUT command that sent the Set Key CbCS page. The following inputs shall be used with the specified integrity check value algorithm:

- a) The contents of the SEED field in the Set Key CbCS page as the string for which the integrity check value is to be computed; and
- b) The generation key component of the master key (see 5.14.6.8.11) for the logical unit that is processing the Set Key CbCS page as the cryptographic key.

#### 7.6.4.5.5 Set Master Key – Seed Exchange CbCS page

The Set Master Key – Seed Exchange CbCS page (see table 450) in a SECURITY PROTOCOL OUT command initiates a CbCS master key update CCS (see 5.14.6.8.11.4) and begins a Diffie-Hellman key exchange protocol as part of that CCS.

**Table 450 — Set Master Key – Seed Exchange CbCS page format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)	PAGE CODE (D010h)						(LSB)
1								
2	(MSB)	PAGE LENGTH (n-3)						(LSB)
3								
4	(MSB)	D-H ALGORITHM						(LSB)
7								
8	(MSB)	D-H DATA LENGTH (n-11)						(LSB)
11								
12	(MSB)	D-H DATA						(LSB)
n								

The PAGE CODE field set to D010h specifies that the Set Master Key – Seed Exchange CbCS page follows.

The PAGE LENGTH field specifies the number of bytes that follow in the Set Master Key – Seed Exchange CbCS page. If the PAGE LENGTH field is set to a value that is smaller than ten, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The D-H ALGORITHM field specifies the Diffie-Hellman (D-H) algorithm to be used in the seed exchange protocol. The value in the D-H ALGORITHM field is selected from the codes that table 88 (see 5.14.8) lists as Diffie-Hellman algorithms with finite field D-H computations. The Diffie-Hellman algorithm selected specifies the DH\_generator value and DH\_prime value to be used in the computation of the D-H DATA field as described in this subclause.

If the value in the D-H ALGORITHM field does not appear in the Supported Diffie-Hellman algorithms list in the Unchangeable CbCS Parameters CbCS page (see 7.6.4.3.3), then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The D-H DATA LENGTH field specifies the number of bytes that follow in D-H DATA field. If the D-H DATA LENGTH field is set to a value that is inconsistent with the Diffie-Hellman algorithm specified by the D-H ALGORITHM field, then the

command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The contents of the D-H DATA field are computed as follows:

- 1) A random number,  $x$ , is generated having a value between 0 and  $DH\_prime$  minus one observing the requirements in RFC 4086; and
- 2) The D-H DATA field is set to  $DH\_generator^x$  modulo  $DH\_prime$ , where the  $DH\_generator$  and  $DH\_prime$  values are identified by the value in the D-H ALGORITHM field.

#### 7.6.4.5.6 Set Master Key – Change Master Key CbCS page

The Set Master Key – Change Master Key CbCS page (see table 451) concludes a CbCS master key update CCS (see 5.14.6.8.11.4) and changes the master key components to the values computed as part of processing completion (see 7.6.4.3.6) for the SECURITY PROTOCOL IN command that transferred the Set Master Key – Seed Exchange CbCS page.

**Table 451 — Set Master Key – Change Master Key CbCS page format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)	PAGE CODE (D011h)						(LSB)
1								
2	(MSB)	PAGE LENGTH (n-3)						(LSB)
3								
4		Reserved						
7								
8	(MSB)	KEY IDENTIFIER						(LSB)
15								
16	(MSB)	APPLICATION CLIENT D-H DATA LENGTH (k-19)						(LSB)
19								
20	(MSB)	APPLICATION CLIENT D-H DATA						(LSB)
k								
k+1	(MSB)	DEVICE SERVER D-H DATA LENGTH (n-k-4)						(LSB)
k+4								
k+5	(MSB)	DEVICE SERVER D-H DATA						(LSB)
n								

The PAGE CODE field set to D011h specifies that the Set Master Key – Change Master Key CbCS page follows.

The PAGE LENGTH field specifies the number of bytes that follow in the Set Master Key – Change Master Key CbCS page. If the PAGE LENGTH field is set to a value that is smaller than 24, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The KEY IDENTIFIER field specifies the value to which the master key CbCS shared key identifier (see 5.14.6.8.11.2) shall be set. If the KEY IDENTIFIER field is set to a value that table 72 (see 5.14.6.8.11.2) describes as reserved in

the CbCS pages that change CbCS shared key values, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The APPLICATION CLIENT D-H DATA LENGTH field specifies the number of bytes that follow in the APPLICATION CLIENT D-H DATA field. If the contents of the APPLICATION CLIENT D-H DATA LENGTH field do not match the number of Diffie-Hellman (D-H) data bytes that the device server received in the SECURITY PROTOCOL OUT command that sent the Set Master Key – Seed Exchange CbCS page (see 7.6.4.5.5) and initiated the current CbCS master key update CCS (see 5.14.6.8.11.4), then the master key shall not be modified and the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The APPLICATION CLIENT D-H DATA field contains Diffie-Hellman data field that was sent in the Set Master Key – Seed Exchange CbCS page that initiated the current CbCS master key update CCS. If the contents of the APPLICATION CLIENT D-H DATA field do not match Diffie-Hellman data that the device server received in the SECURITY PROTOCOL OUT command that sent the Set Master Key – Seed Exchange CbCS page that initiated the current CbCS master key update CCS (see 5.14.6.8.11.4), then the master key shall not be modified and the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The DEVICE SERVER D-H DATA LENGTH field specifies the number of bytes that follow in the DEVICE SERVER D-H DATA field. If the contents of the DEVICE SERVER D-H DATA LENGTH field do not match the number of Diffie-Hellman data bytes that the device server returned in the SECURITY PROTOCOL IN command that returned the Set Master Key – Seed Exchange CbCS page (see 7.6.4.3.6) in the current CbCS master key update CCS, then the master key shall not be modified and the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The DEVICE SERVER D-H DATA field contains Diffie-Hellman data field that was returned in the Set Master Key – Seed Exchange CbCS page that the device server returned in response to a SECURITY PROTOCOL IN COMMAND as part of the current CbCS master key update CCS. If the contents of the DEVICE SERVER D-H DATA field do not match Diffie-Hellman data that the device server sent in the SECURITY PROTOCOL IN command that sent the Set Master Key – Seed Exchange CbCS page, then the master key shall not be modified and the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

## 7.7 Vital product data parameters

### 7.7.1 Vital product data parameters overview and page codes

This subclause describes the vital product data (VPD) page structure and the VPD pages (see table 452) that are applicable to all SCSI devices. These VPD pages are returned by an INQUIRY command with the EVPD bit set to one (see 6.4) and contain vendor specific product information about a logical unit and SCSI target device. The vital product data may include vendor identification, product identification, unit serial numbers, device operating definitions, manufacturing data, field replaceable unit information, and other vendor specific information. This standard defines the structure of the vital product data, but not the contents.

**Table 452 — Vital product data page codes**

Page code	VPD Page Name	Reference	Support Requirements
01h to 7Fh	ASCII Information	7.7.2	Optional
89h	ATA Information	SAT-2	Optional <sup>a</sup>
83h	Device Identification	7.7.3	Mandatory
86h	Extended INQUIRY Data	7.7.4	Optional
85h	Management Network Addresses	7.7.5	Optional
87h	Mode Page Policy	7.7.6	Optional
81h	Obsolete	3.3.7	
82h	Obsolete	3.3.7	
88h	SCSI Ports	7.7.7	Optional
90h	Protocol Specific Logical Unit Information	7.7.8	Protocol specific <sup>b</sup>
91h	Protocol Specific Port Information	7.7.9	Protocol specific <sup>b</sup>
84h	Software Interface Identification	7.7.10	Optional
00h	Supported VPD Pages	7.7.11	Mandatory
80h	Unit Serial Number	7.7.12	Optional
8Ah to 8Fh	Reserved		
92h to AFh	Reserved		
B0h to BFh	(See specific device type)		
C0h to FFh	Vendor specific		
Annex D contains a listing of VPD page codes in numeric order.			
<sup>a</sup> See SAT-2 for support requirements.			
<sup>b</sup> See applicable SCSI transport protocol standard (see 3.1.140) for support requirements.			

### 7.7.2 ASCII Information VPD page

The ASCII Information VPD page (see table 453) contains information for the field replaceable unit code returned in the sense data (see 4.5).

**Table 453 — ASCII Information VPD page**

Bit Byte	7	6	5	4	3	2	1	0	
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE					
1	PAGE CODE (01h to 7Fh)								
2	Reserved								
3	PAGE LENGTH (n-3)								
4	ASCII LENGTH (m-4)								
5	(MSB)								
m				ASCII INFORMATION					(LSB)
m+1									
n	Vendor specific information								

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field are defined in 6.4.2.

The PAGE CODE field contains the same value as in the PAGE OR OPERATION CODE field of the INQUIRY CDB (see 6.4) and is associated with the FIELD REPLACEABLE UNIT CODE field returned in the sense data.

NOTE 70 - The FIELD REPLACEABLE UNIT CODE field in the sense data provides for 255 possible codes, while the PAGE CODE field provides for only 127 possible codes. For that reason it is not possible to return ASCII Information VPD pages for the upper code values.

The PAGE LENGTH field indicates the length of the following VPD page data. The relationship between the PAGE LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The ASCII LENGTH field indicates the length in bytes of the ASCII INFORMATION field that follows. A value of zero in this field indicates that no ASCII information is available for the indicated page code. The relationship between the ASCII LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The ASCII INFORMATION field contains ASCII information concerning the field replaceable unit identified by the page code. The data in this field shall be formatted in one or more character string lines. Each line shall contain only graphic codes (i.e., code values 20h through 7Eh) and shall be terminated with a NULL (00h) character.

The contents of the vendor specific information field is not defined in this standard.



### 7.7.3 Device Identification VPD page

#### 7.7.3.1 Device Identification VPD page overview

The Device Identification VPD page (see table 454) provides the means to retrieve designation descriptors applying to the logical unit. Logical units may have more than one designation descriptor (e.g., if several types or associations of designator are supported). Designators consist of one or more of the following:

- a) Logical unit names;
- b) SCSI target port identifiers;
- c) SCSI target port names;
- d) SCSI device names;
- e) Relative target port identifiers;
- f) Primary target port group number; or
- g) Logical unit group number.

Designation descriptors shall be assigned to the peripheral device (e.g., a disk drive) and not to the currently mounted media, in the case of removable media devices. Operating systems are expected to use the designation descriptors during system configuration activities to determine whether alternate paths exist for the same peripheral device.

**Table 454 — Device Identification VPD page**

Bit Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (83h)							
2	(MSB)							
3	PAGE LENGTH (n-3)							(LSB)
	Designation descriptor list							
4	Designation descriptor [first]							
	⋮							
n	Designation descriptor [last]							

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field in table 454 are as defined in 6.4.2.

The PAGE LENGTH field indicates the length of the designation descriptor list. The relationship between the PAGE LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

Each designation descriptor (see table 455) contains information identifying the logical unit, SCSI target device containing the logical unit, or access path (i.e., target port) used by the command and returned parameter data. The Device Identification VPD page shall contain the designation descriptors enumerated in 7.7.3.2.

**Table 455 — Designation descriptor**

Bit Byte	7	6	5	4	3	2	1	0
0	PROTOCOL IDENTIFIER				CODE SET			
1	PIV	Reserved	ASSOCIATION		DESIGNATOR TYPE			
2	Reserved							
3	DESIGNATOR LENGTH (n-3)							
4	DESIGNATOR							
n								

The PROTOCOL IDENTIFIER field may indicate the SCSI transport protocol to which the designation descriptor applies. If the ASSOCIATION field contains a value other than 01b (i.e., target port) or 10b (i.e., SCSI target device) or the PIV bit is set to zero, then the PROTOCOL IDENTIFIER field contents are reserved. If the ASSOCIATION field contains a value of 01b or 10b and the PIV bit is set to one, then the PROTOCOL IDENTIFIER field shall contain one of the values shown in table 361 (see 7.5.1) to indicate the SCSI transport protocol to which the designation descriptor applies.

The CODE SET field contains a code set enumeration (see 4.4.3) that indicates the format of the DESIGNATOR field.

A protocol identifier valid (PIV) bit set to zero indicates the PROTOCOL IDENTIFIER field contents are reserved. If the ASSOCIATION field contains a value of 01b or 10b, then a PIV bit set to one indicates the PROTOCOL IDENTIFIER field contains a valid protocol identifier selected from the values shown in table 361 (see 7.5.1). If the ASSOCIATION field contains a value other than 01b or 10b, then the PIV bit contents are reserved.

The ASSOCIATION field indicates the entity with which the DESIGNATOR field is associated, as described in table 456. If a logical unit returns a designation descriptor with the ASSOCIATION field set to 00b or 10b, it shall return the same descriptor when it is accessed through any other I\_T nexus.

**Table 456 — ASSOCIATION field**

Code	Description
00b	The DESIGNATOR field is associated with the addressed logical unit.
01b	The DESIGNATOR field is associated with the target port that received the request.
10b	The DESIGNATOR field is associated with the SCSI target device that contains the addressed logical unit.
11b	Reserved

The DESIGNATOR TYPE field (see table 457) indicates the format and assignment authority for the designator.

**Table 457 — DESIGNATOR TYPE field**

Code	Description	Reference
0h	Vendor specific	7.7.3.3
1h	T10 vendor ID based	7.7.3.4
2h	EUI-64 based	7.7.3.5
3h	NAA	7.7.3.6
4h	Relative target port identifier	7.7.3.7
5h	Target port group	7.7.3.8
6h	Logical unit group	7.7.3.9
7h	MD5 logical unit identifier	7.7.3.10
8h	SCSI name string	7.7.3.11
9h to Fh	Reserved	

The DESIGNATOR LENGTH field indicates the length in bytes of the DESIGNATOR field. The relationship between the DESIGNATOR LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The DESIGNATOR field contains the designator as described by the ASSOCIATION, DESIGNATOR TYPE, CODE SET, and DESIGNATOR LENGTH fields.

### 7.7.3.2 Device designation descriptor requirements

#### 7.7.3.2.1 Designation descriptors for logical units other than well known logical units

For each logical unit that is not a well known logical unit, the Device Identification VPD page shall include at least one designation descriptor in which a logical unit name (see SAM-4) is indicated. The designation descriptor shall have the ASSOCIATION field set to 00b (i.e., logical unit) and the DESIGNATOR TYPE field set to:

- a) 1h (i.e., T10 vendor ID based);
- b) 2h (i.e., EUI-64-based);
- c) 3h (i.e., NAA); or
- d) 8h (i.e., SCSI name string).

At least one designation descriptor should have the DESIGNATOR TYPE field set to:

- a) 2h (i.e., EUI-64-based);
- b) 3h (i.e., NAA); or
- c) 8h (i.e., SCSI name string).

In the case of virtual logical units (e.g., volume sets as defined by SCC-2), designation descriptors should contain a DESIGNATOR TYPE field set to:

- a) 2h (i.e., EUI-64-based);
- b) 3h (i.e., NAA); or
- c) 8h (i.e., SCSI name string).

In the case of virtual logical units that have an EUI-64 based designation descriptor (see 7.7.3.5) the DESIGNATOR LENGTH field should be set to:

- a) 0Ch (i.e., EUI-64-based 12-byte); or
- b) 10h (i.e., EUI-64-based 16-byte).

In the case of virtual logical units that have an NAA designation descriptor (see 7.7.3.6) the NAA field should be set to 6h (i.e., IEEE Registered Extended).

The Device Identification VPD page shall contain the same set of designation descriptors with the ASSOCIATION field set to 00b (i.e., logical unit) regardless of the I\_T nexus being used to retrieve the designation descriptors.

For logical units that are not well known logical units, the requirements for SCSI target device designation descriptors are defined in 7.7.3.2.4 and the requirements for SCSI target port designation descriptors are defined in 7.7.3.2.3.

#### **7.7.3.2.2 Designation descriptors for well known logical units**

Well known logical units shall not return any designation descriptors with the ASSOCIATION field set to 00b (i.e., logical unit).

The Device Identification VPD page shall contain the same set of designation descriptors with the ASSOCIATION field set to 10b (i.e., SCSI target device) regardless of the I\_T nexus being used to retrieve the designation descriptors.

#### **7.7.3.2.3 Designation descriptors for SCSI target ports**

##### **7.7.3.2.3.1 Relative target port identifiers**

For the target port through which the Device Identification VPD page is accessed, the Device Identification VPD page should include one designation descriptor with the ASSOCIATION field set to 01b (i.e., target port) and the DESIGNATOR TYPE field set to 4h (i.e., relative target port identifier) identifying the target port being used to retrieve the designation descriptors.

##### **7.7.3.2.3.2 Target port names or identifiers**

For the SCSI target port through which the Device Identification VPD page is accessed, the Device Identification VPD page should include one designation descriptor in which the target port name or identifier (see SAM-4) is indicated. The designation descriptor, if any, shall have the ASSOCIATION field set to 01b (i.e., target port) and the DESIGNATOR TYPE field set to:

- a) 2h (i.e., EUI-64-based);
- b) 3h (i.e., NAA); or
- c) 8h (i.e., SCSI name string).

If the SCSI transport protocol standard (see 3.1.140) for the target port defines target port names, the designation descriptor, if any, shall contain the target port name. If the SCSI transport protocol for the target port does not define target port names, the designation descriptor, if any, shall contain the target port identifier.

#### **7.7.3.2.4 Designation descriptors for SCSI target devices**

If the SCSI target device contains a well known logical unit, the Device Identification VPD page shall have one or more designation descriptors for the SCSI target device. If the SCSI target device does not contain a well known logical unit, the Device Identification VPD page should have one or more designation descriptors for the SCSI target device.

Each SCSI target device designation descriptor, if any, shall have the ASSOCIATION field set to 10b (i.e., SCSI target device) and the DESIGNATOR TYPE field set to:

- a) 2h (i.e., EUI-64-based);
- b) 3h (i.e., NAA); or
- c) 8h (i.e., SCSI name string).

The Device Identification VPD page shall contain designation descriptors, if any, for all the SCSI device names for all the SCSI transport protocols supported by the SCSI target device.

### 7.7.3.3 Vendor specific designator format

If the designator type is 0h (i.e., vendor specific), no assignment authority was used and there is no guarantee that the designator is globally unique (i.e., the identifier is vendor specific). Table 458 defines the DESIGNATOR field format.

**Table 458 — Vendor specific DESIGNATOR field format**

Bit Byte	7	6	5	4	3	2	1	0
0	VENDOR SPECIFIC IDENTIFIER							
n								

### 7.7.3.4 T10 vendor ID based designator format

If the designator type is 1h (i.e., T10 vendor ID based), the DESIGNATOR field has the format shown in table 459.

**Table 459 — T10 vendor ID based DESIGNATOR field format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) T10 VENDOR IDENTIFICATION (LSB)							
7								
8	VENDOR SPECIFIC IDENTIFIER							
n								

The T10 VENDOR IDENTIFICATION field contains eight bytes of left-aligned ASCII data (see 4.4.1) identifying the vendor of the product. The data shall be left aligned within this field. The T10 vendor identification shall be one assigned by INCITS. A list of assigned T10 vendor identifications is in Annex E and on the T10 web site (<http://www.T10.org>).

NOTE 71 - The T10 web site (<http://www.t10.org>) provides a convenient means to request an identification code.

The organization associated with the T10 vendor identification is responsible for ensuring that the VENDOR SPECIFIC DESIGNATOR field is unique in a way that makes the entire DESIGNATOR field unique. A recommended method of constructing a unique DESIGNATOR field is to concatenate the PRODUCT IDENTIFICATION field from the standard INQUIRY data (see 6.4.2) and the PRODUCT SERIAL NUMBER field from the Unit Serial Number VPD page (see 7.7.12).

### 7.7.3.5 EUI-64 based designator format

#### 7.7.3.5.1 EUI-64 based designator format overview

If the designator type is 2h (i.e., EUI-64 based identifier), the DESIGNATOR LENGTH field (see table 460) indicates the format of the designation descriptor.

**Table 460 — EUI-64 based designator lengths**

Designator Length	Description	Reference
08h	EUI-64 identifier	7.7.3.5.2
0Ch	EUI-64 based 12-byte identifier	7.7.3.5.3
10h	EUI-64 based 16-byte identifier	7.7.3.5.4
All other values	Reserved	

#### 7.7.3.5.2 EUI-64 designator format

If the designator type is 2h (i.e., EUI-64 based identifier) and the DESIGNATOR LENGTH field is set to 08h, the DESIGNATOR field has the format shown in table 461. The CODE SET field shall be set to 1h (i.e., binary).

**Table 461 — EUI-64 DESIGNATOR field format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) IEEE COMPANY_ID (LSB)							
2								
3	(MSB) VENDOR SPECIFIC EXTENSION IDENTIFIER (LSB)							
7								

The IEEE COMPANY\_ID field contains a 24 bit OUI (see 3.1.97) assigned by the IEEE.

The VENDOR SPECIFIC EXTENSION IDENTIFIER field contains a 40 bit numeric value that is uniquely assigned by the organization associated with the IEEE company\_id as required by the IEEE definition of EUI-64 (see 3.1.52).

### 7.7.3.5.3 EUI-64 based 12-byte designator format

If the designator type is 2h (i.e., EUI-64 based identifier) and the DESIGNATOR LENGTH field is set to 0Ch, the DESIGNATOR field has the format shown in table 462. The CODE SET field shall be set to 1h (i.e., binary).

**Table 462 — EUI-64 based 12-byte DESIGNATOR field format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____							
2	IEEE COMPANY_ID (LSB)							
3	(MSB) _____							
7	VENDOR SPECIFIC EXTENSION IDENTIFIER (LSB)							
8	(MSB) _____							
11	DIRECTORY ID (LSB)							

The IEEE COMPANY\_ID field and VENDOR SPECIFIC EXTENSION IDENTIFIER field are defined in 7.7.3.5.2.

The DIRECTORY ID field contains a directory identifier, as specified by ISO/IEC 13213:1994.

NOTE 72 - The EUI-64 based 12 byte format may be used to report IEEE 1394 target port identifiers (see SBP-3).

### 7.7.3.5.4 EUI-64 based 16-byte designator format

If the designator type is 2h (i.e., EUI-64 based identifier) and the DESIGNATOR LENGTH field is set to 10h, the DESIGNATOR field has the format shown in table 463. The CODE SET field shall be set to 1h (i.e., binary).

**Table 463 — EUI-64 based 16-byte DESIGNATOR field format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____							
7	IDENTIFIER EXTENSION (LSB)							
8	(MSB) _____							
10	IEEE COMPANY_ID (LSB)							
11	(MSB) _____							
15	VENDOR SPECIFIC EXTENSION IDENTIFIER (LSB)							

The IDENTIFIER EXTENSION field contains a 64 bit numeric value.

The IEEE COMPANY\_ID field and VENDOR SPECIFIC EXTENSION IDENTIFIER field are defined in 7.7.3.5.2.

NOTE 73 - The EUI-64 based 16-byte format may be used to report SCSI over RDMA target port identifiers (see SRP).

### 7.7.3.6 NAA designator format

#### 7.7.3.6.1 NAA identifier basic format

If the designator type is 3h (i.e., NAA identifier), the DESIGNATOR field has the format shown in table 464. This format is compatible with the Name\_Identifier format defined in FC-FS.

**Table 464 — NAA DESIGNATOR field format**

Bit Byte	7	6	5	4	3	2	1	0
0	NAA				NAA specific data			
1								
n								

The Name Address Authority (NAA) field (see table 465) defines the format of the NAA specific data in the designator.

**Table 465 — Name Address Authority (NAA) field**

Code	Description	Reference
2h	IEEE Extended	7.7.3.6.2
3h	Locally Assigned	7.7.3.6.3
5h	IEEE Registered	7.7.3.6.4
6h	IEEE Registered Extended	7.7.3.6.5
All others	Reserved	

#### 7.7.3.6.2 NAA IEEE Extended designator format

If NAA is 2h (i.e., IEEE Extended), the eight byte fixed length DESIGNATOR field shall have the format shown in table 466. The CODE SET field shall be set to 1h (i.e., binary) and the DESIGNATOR LENGTH field shall be set to 08h.

**Table 466 — NAA IEEE Extended DESIGNATOR field format**

Bit Byte	7	6	5	4	3	2	1	0
0	NAA (2h)				(MSB)			
1	VENDOR SPECIFIC IDENTIFIER A							(LSB)
2	(MSB)	IEEE COMPANY_ID						
4								(LSB)
5	(MSB)	VENDOR SPECIFIC IDENTIFIER B						
7								(LSB)

The IEEE COMPANY\_ID field contains a 24 bit canonical form OUI (see 3.1.97) assigned by the IEEE.

The VENDOR SPECIFIC IDENTIFIER A contains a 12 bit numeric value that is uniquely assigned by the organization associated with the IEEE company\_id.



The VENDOR SPECIFIC IDENTIFIER B contains a 24 bit numeric value that is uniquely assigned by the organization associated with the IEEE company\_id.

NOTE 74 - The EUI-64 format includes a 40 bit vendor specific identifier. The IEEE Extended format includes 36 bits vendor specific identifier in two fields.

### 7.7.3.6.3 NAA Locally Assigned designator format

If NAA is 3h (i.e., Locally Assigned), the eight byte fixed length DESIGNATOR field shall have the format shown in table 467. The CODE SET field shall be set to 1h (i.e., binary) and the DESIGNATOR LENGTH field shall be set to 08h.

**Table 467 — NAA Locally Assigned DESIGNATOR field format**

Bit Byte	7	6	5	4	3	2	1	0
0	NAA (3h)				LOCALLY ADMINISTERED VALUE			
1								
7								

The LOCALLY ADMINISTERED VALUE field contains a 60 bit value that is assigned by an administrator to be unique within the set of SCSI domains that are accessible by a common instance of an administrative tool or tools.

### 7.7.3.6.4 NAA IEEE Registered designator format

If NAA is 5h (i.e., IEEE Registered), the eight byte fixed length DESIGNATOR field shall have the format shown in table 468. The CODE SET field shall be set to 1h (i.e., binary) and the DESIGNATOR LENGTH field shall be set to 08h.

**Table 468 — NAA IEEE Registered DESIGNATOR field format**

Bit Byte	7	6	5	4	3	2	1	0
0	NAA (5h)				(MSB)			
1	IEEE COMPANY_ID							
2								
3								
4	VENDOR SPECIFIC IDENTIFIER							
7								

The IEEE COMPANY\_ID field contains a 24 bit canonical form OUI (see 3.1.97) assigned by the IEEE.

The VENDOR SPECIFIC IDENTIFIER a 36 bit numeric value that is uniquely assigned by the organization associated with the IEEE company\_id.

NOTE 75 - The EUI-64 identifier includes a 40 bit vendor specific identifier. The IEEE Registered format includes a 36 bit vendor specific identifier.

### 7.7.3.6.5 NAA IEEE Registered Extended designator format

If NAA is 6h (i.e., IEEE Registered Extended), the sixteen byte fixed length DESIGNATOR field shall have the format shown in table 469. The CODE SET field shall be set to 1h (i.e., binary) and the DESIGNATOR LENGTH field shall be set to 10h.

**Table 469 — NAA IEEE Registered Extended DESIGNATOR field format**

Bit Byte	7	6	5	4	3	2	1	0
0	NAA (6h)				(MSB)			
1	IEEE COMPANY_ID							
2								
3								
4	VENDOR SPECIFIC IDENTIFIER							
7								
8	(MSB)	VENDOR SPECIFIC IDENTIFIER EXTENSION						
15								(LSB)

The IEEE COMPANY\_ID field contains a 24 bit canonical form OUI (see 3.1.97) assigned by the IEEE.

The VENDOR SPECIFIC IDENTIFIER field contains a 36 bit numeric value that is uniquely assigned by the organization associated with the IEEE company\_id.

NOTE 76 - The EUI-64 format includes a 40 bit vendor specific identifier. The IEEE Registered Extended format includes a 36 bit vendor specific identifier.

The VENDOR SPECIFIC IDENTIFIER EXTENSION a 64 bit numeric value that is assigned to make the DESIGNATOR field unique.

### 7.7.3.7 Relative target port designator format

If the designator type is 4h (i.e., relative target port identifier) and the ASSOCIATION field contains 01b (i.e., target port), then the DESIGNATOR field shall have the format shown in table 470. The CODE SET field shall be set to 1h (i.e., binary) and the DESIGNATOR LENGTH field shall be set to 04h. If the ASSOCIATION field does not contain 01b, use of this designator type is reserved.

**Table 470 — Relative target port DESIGNATOR field format**

Bit Byte	7	6	5	4	3	2	1	0
0	Obsolete							
1								
2	(MSB)	RELATIVE TARGET PORT IDENTIFIER						
3								(LSB)

The RELATIVE TARGET PORT IDENTIFIER field (see table 471) contains the relative port identifier (see 3.1.120) of the target port on which the INQUIRY command was received.

**Table 471 — RELATIVE TARGET PORT IDENTIFIER field**

Code	Description
0h	Reserved
1h	Relative port 1, historically known as port A
2h	Relative port 2, historically known as port B
3h to FFFFh	Relative port 3 through 65 535

### 7.7.3.8 Target port group designator format

If the designator type is 5h (i.e., target port group) and the ASSOCIATION value is 01b (i.e., target port), the four byte fixed length DESIGNATOR field shall have the format shown in table 472. The CODE SET field shall be set to 1h (i.e., binary) and the DESIGNATOR LENGTH field shall be set to 04h. If the ASSOCIATION field does not contain 01b, use of this designator type is reserved.

**Table 472 — Target port group DESIGNATOR field format**

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved							
1								
2	(MSB)	TARGET PORT GROUP						
3								(LSB)

The TARGET PORT GROUP field indicates the primary target port group to which the target port is a member (see 5.9).

### 7.7.3.9 Logical unit group designator format

A logical unit group is a group of logical units that share the same primary target port group (see 5.9) definitions. The primary target port groups maintain the same primary target port group asymmetric access states for all logical units in the same logical unit group. A logical unit shall be in no more than one logical unit group.

If the designator type is 6h (i.e., logical unit group) and the ASSOCIATION value is 00b (i.e., logical unit), the four byte fixed length DESIGNATOR field shall have the format shown in table 473. The CODE SET field shall be set to 1h (i.e., binary) and the DESIGNATOR LENGTH field shall be set to 04h. If the ASSOCIATION field does not contain 00b, use of this designator type is reserved.

**Table 473 — Logical unit group DESIGNATOR field format**

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved							
1								
2	(MSB)	LOGICAL UNIT GROUP						
3								(LSB)

The LOGICAL UNIT GROUP field indicates the logical unit group to which the logical unit is a member.

### 7.7.3.10 MD5 logical unit designator format

If the designator type is 7h (i.e., MD5 logical unit identifier) and the ASSOCIATION value is 00b (i.e., logical unit), the DESIGNATOR field has the format shown in table 474. The CODE SET field shall be set to 1h (i.e., binary). The MD5 logical unit designator shall not be used if a logical unit provides unique identification using designator types 2h (i.e., EUI-64 based identifier), 3h (i.e., NAA identifier), or 8h (i.e., SCSI name string). A bridge device may return a MD5 logical unit designator type for that logical unit that does not support the Device Identification VPD page (see 7.7.3).

If the ASSOCIATION field does not contain 00b, use of this designator type is reserved.

**Table 474 — MD5 logical unit DESIGNATOR field format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
15	MD5 LOGICAL UNIT IDENTIFIER (LSB)							

The MD5 LOGICAL UNIT IDENTIFIER field contains the message digest of the supplied message input. The message digest shall be generated using the MD5 message-digest algorithm as specified in RFC 1321 (see 2.5) with the following information as message input:

- 1) The contents of the T10 VENDOR IDENTIFICATION field in the standard INQUIRY data (see 6.4.2);
- 2) The contents of the PRODUCT IDENTIFICATION field in the standard INQUIRY data;
- 3) The contents of the PRODUCT SERIAL NUMBER field in the Unit Serial Number VPD page (see 7.7.12);
- 4) The contents of a vendor specific DESIGNATOR field (designator type 0h) from the Device Identification VPD page; and
- 5) The contents of a T10 vendor ID based DESIGNATOR field (designator type 1h) from the Device Identification VPD page.

If a field or page is not available, the message input for that field or page shall be 8 bytes of ASCII space characters (i.e., 20h).

The uniqueness of the MD5 logical unit identifier is dependent upon the relative degree of randomness (i.e., the entropy) of the message input. If it is found that two or more logical units have the same MD5 logical unit identifier, the application client should determine in a vendor specific manner whether the logical units are the same entities.

The MD5 logical unit identifier example described in this paragraph and shown in table 475 and table 476 is not a normative part of this standard. The data available for input to the MD5 algorithm for this example is shown in table 475.

**Table 475 — MD5 logical unit identifier example available data**

MD5 message input	Available	Contents
T10 VENDOR IDENTIFICATION field	Yes	T10
PRODUCT IDENTIFICATION field	Yes	MD5 Logical Unit
PRODUCT SERIAL NUMBER field	Yes	01234567
vendor specific DESIGNATOR field	No	
T10 vendor ID based DESIGNATOR field	No	

The concatenation of the fields in table 475 to form input to the MD5 algorithm is shown in table 476.

**Table 476 — Example MD5 input for computation of a logical unit identifier**

Bytes	Hexadecimal values				ASCII values
00 to 15	54 31 30 20	20 20 20 20	4D 44 35 20	4C 6F 67 69	T10 MD5 Logi
16 to 31	63 61 6C 20	55 6E 69 74	30 31 32 33	34 35 36 37	cal Unit01234567
32 to 47	20 20 20 20	20 20 20 20	20 20 20 20	20 20 20 20	
NOTE 1 Non-printing ASCII characters are shown as '.'.					

Based on the example inputs shown in table 475 and the concatenation of the inputs shown in table 476, the MD5 base 16 algorithm described in RFC 1321 produces the value 8FAC A22A 0AC0 3839 1255 25F2 0EFE 2E7Eh.

### 7.7.3.11 SCSI name string designator format

If the designator type is 8h (i.e., SCSI name string), the DESIGNATOR field has the format shown in table 477. The CODE SET field shall be set to 3h (i.e., UTF-8).

**Table 477 — SCSI name string DESIGNATOR field format**

Bit Byte	7	6	5	4	3	2	1	0
0	SCSI NAME STRING							
n								

The null-terminated, null-padded (see 4.4.2) SCSI NAME STRING field contains a UTF-8 format string. The number of bytes in the SCSI NAME STRING field (i.e., the value in the DESIGNATOR LENGTH field) shall be no larger than 256 and shall be a multiple of four.

The SCSI NAME STRING field starts with either:

- The four UTF-8 characters "eui." concatenated with 16, 24, or 32 hexadecimal digits (i.e., the UTF-8 characters 0 through 9 and A through F) for an EUI-64 based identifier (see 7.7.3.5). The first hexadecimal digit shall be the most significant four bits of the first byte (i.e., most significant byte) of the EUI-64 based identifier;
- The four UTF-8 characters "naa." concatenated with 16 or 32 hexadecimal digits for an NAA identifier (see 7.7.3.6). The first hexadecimal digit shall be the most significant four bits of the first byte (i.e., most significant byte) of the NAA identifier; or
- The four UTF-8 characters "iqn." concatenated with an iSCSI Name for an iSCSI-name based identifier (see iSCSI).

If the ASSOCIATION field is set to 00b (i.e., logical unit) and the SCSI NAME STRING field starts with the four UTF-8 characters "iqn.", the SCSI NAME STRING field ends with the five UTF-8 characters ",L,0x" concatenated with 16 hexadecimal digits for the logical unit name extension. The logical unit name extension is a UTF-8 string containing no more than 16 hexadecimal digits. The logical unit name extension is assigned by the SCSI target device vendor and shall be assigned so the logical unit name is worldwide unique.

If the ASSOCIATION field is set to 01b (i.e., target port), the SCSI NAME STRING field ends with the five UTF-8 characters ",t,0x" concatenated with two or more hexadecimal digits as specified in the applicable SCSI transport protocol standard (see 3.1.140).

If the ASSOCIATION field is set to 10b (i.e., SCSI target device), the SCSI NAME STRING field has no additional characters.

NOTE 77 - The notation used in this subclause to specify exact UTF-8 character strings is described in 3.6.1.

#### 7.7.4 Extended INQUIRY Data VPD page

The Extended INQUIRY Data VPD page (see table 478) provides the application client with a means to obtain information about the logical unit.

**Table 478 — Extended INQUIRY Data VPD page**

Bit Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (86h)							
2	Reserved							
3	PAGE LENGTH (3Ch)							
4	Reserved		SPT			GRD_CHK	APP_CHK	REF_CHK
5	Reserved		UASK_SUP	GROUP_SUP	PRIOR_SUP	HEADSUP	ORDSUP	SIMPSUP
6	Reserved				WU_SUP	CRD_SUP	NV_SUP	V_SUP
7	Reserved							LUICLR
8	Reserved							CBCS
9	Reserved				MULTI I_T NEXUS MICROCODE DOWNLOAD			
10	Reserved							
63								

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field are as defined in 6.4.2.

The PAGE LENGTH field indicates the length of the following VPD page data and shall be set to 60. The relationship between the PAGE LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

A supported protection type (SPT) field (see table 479) indicates the type of protection the logical unit supports. The SPT field shall be ignored if the PROTECT bit (see 6.4.2) is set to zero.

**Table 479 — Supported protection type (SPT) field**

Code	Definition
000b	The logical unit supports type 1 protection (see SBC-3)
001b	The logical unit supports type 1 and type 2 protection (see SBC-3)
010b	Reserved
011b	The logical unit supports type 1 and type 3 protection (see SBC-3)
100b to 111b	Reserved

A guard check (GRD\_CHK) bit set to zero indicates that the device server does not check the LOGICAL BLOCK GUARD field in the protection information (see SBC-2), if any. A GRD\_CHK bit set to one indicates that the device server checks the LOGICAL BLOCK GUARD field in the protection information, if any.

An application tag check (APP\_CHK) bit set to zero indicates that the device server does not check the LOGICAL BLOCK APPLICATION TAG field in the protection information (see SBC-2), if any. An APP\_CHK bit set to one indicates that the device server checks the LOGICAL BLOCK APPLICATION TAG field in the protection information, if any.

A reference tag check (REF\_CHK) bit set to zero indicates that the device server does not check the LOGICAL BLOCK REFERENCE TAG field in the protection information (see SBC-2), if any. A REF\_CHK bit set to one indicates that the device server checks the LOGICAL BLOCK REFERENCE TAG field in the protection information, if any.

A unit attention condition sense key specific data supported (UASK\_SUP) bit set to one indicates that the device server returns sense-key specific data for the UNIT ATTENTION sense key (see 4.5.2.4.6). A UASK\_SUP bit set to zero indicates that the device server does not return sense-key specific data for the UNIT ATTENTION sense key.

A grouping function supported (GROUP\_SUP) bit set to one indicates that the grouping function (see SBC-2) is supported by the device server. A GROUP\_SUP bit set to zero indicates that the grouping function is not supported.

A priority supported (PRIOR\_SUP) bit set to one indicates that command priority (see SAM-4) is supported by the logical unit. A PRIOR\_SUP bit set to zero indicates that command priority is not supported.

A head of queue supported (HEADSUP) bit set to one indicates that the HEAD OF QUEUE task attribute (see SAM-4) is supported by the logical unit. A HEADSUP bit set to zero indicates that the HEAD OF QUEUE task attribute is not supported. If the HEADSUP bit is set to zero, application clients should not specify the HEAD OF QUEUE task attribute as an Execute Command (see 4.2) procedure call argument.

An ordered supported (ORDSUP) bit set to one indicates that the ORDERED task attribute (see SAM-4) is supported by the logical unit. An ORDSUP bit set to zero indicates that the ORDERED task attribute is not supported. If the ORDSUP bit is set to zero, application clients should not specify the ORDERED task attribute as an Execute Command procedure call argument.

The simple supported (SIMPSUP) bit shall be set to one indicating that the SIMPLE task attribute (see SAM-4) is supported by the logical unit.

SAM-4 defines how unsupported task attributes are processed.

A write uncorrectable supported (WU\_SUP) bit set to zero indicates that the device server does not support application clients setting the WR\_UNCOR bit to one in the WRITE LONG command (see SBC-3). A WU\_SUP bit set to one indicates that the device server supports application clients setting the WR\_UNCOR bit to one in the WRITE LONG command.

A correction disable supported (CRD\_SUP) bit set to zero indicates that the device server does not support application clients setting the COR\_DIS bit to one in the WRITE LONG command (see SBC-3). A CRD\_SUP bit set to one indicates that the device server supports application clients setting the COR\_DIS bit to one in the WRITE LONG command.

A non-volatile cache supported (NV\_SUP) bit set to one indicates that the device server supports a non-volatile cache (see 3.1.91) and that the applicable command standard (see 3.1.27) defines features using this cache (e.g., the FUA\_NV bit in SBC-2). An NV\_SUP bit set to zero indicates that the device server may or may not support a non-volatile cache.

A volatile cache supported (V\_SUP) bit set to one indicates that the device server supports a volatile cache (see 3.1.182) and that the applicable command standard (see 3.1.27) defines features using this cache (e.g., the FUA bit in SBC-2). An V\_SUP bit set to zero indicates that the device server may or may not support a volatile cache.

A logical unit I\_T nexus clear (LUICLR) bit set to one indicates the SCSI target device clears any unit attention condition with an additional sense code of REPORTED LUNS DATA HAS CHANGED in each logical unit acces-

sible to an I\_T nexus after reporting the unit attention condition for any logical unit over that I\_T nexus (see SAM-4). An LUICLR bit set to zero indicates the SCSI target device clears unit attention conditions according to a previous version of this standard. The LUICLR bit shall be set to one.

A capability-based command security (CBCS) bit set to one indicates that the logical unit supports the capability-based command security technique (see 5.14.6.8). A CBCS bit set to zero indicates that the logical unit does not support the capability-based command security technique.

The MULTI I\_T NEXUS MICROCODE DOWNLOAD field (see table 92 in 5.16) indicates how the device server handles concurrent attempts to download microcode using the WRITE BUFFER command (see 5.16) from multiple I\_T nexuses.

7.7.5 Management Network Addresses VPD page

The Management Network Addresses VPD page (see table 480) provides a list of network addresses of management services associated with a SCSI target device, target port, or logical unit.

Table 480 — Management Network Addresses VPD page

Bit Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (85h)							
2	(MSB) _____							
3	PAGE LENGTH (n-3) _____ (LSB)							
	Network services descriptor list							
4	_____							
	Network services descriptor [first] _____							
	⋮							
	_____							
n	Network services descriptor [last] _____							

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field are as defined in 6.4.2.

The PAGE LENGTH field indicates the length of the network services descriptor list. The relationship between the PAGE LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.



Each network service descriptor (see table 481) contains information about one management service.

**Table 481 — Network service descriptor format**

Bit Byte	7	6	5	4	3	2	1	0	
0	Reserved	ASSOCIATION		SERVICE TYPE					
1	Reserved								
2	(MSB)	NETWORK ADDRESS LENGTH (n-3)							(LSB)
3									
4									
n	NETWORK ADDRESS								

The ASSOCIATION field (see table 456 in 7.7.3.1) indicates the entity (i.e., SCSI target device, target port, or logical unit) with which the service is associated.

The SERVICE TYPE field (see table 482) allows differentiation of multiple services with the same protocol running at different port numbers or paths.

NOTE 78 - A SCSI target device may provide separate HTTP services for configuration and diagnostics. One of these services may use the standard HTTP port 80 (see 3.1.174) and the other service may use a different port (e.g., 8080).

**Table 482 — SERVICE TYPE field**

Code	Description
00h	Unspecified
01h	Storage Configuration Service
02h	Diagnostics
03h	Status
04h	Logging
05h	Code Download
06h to 1Fh	Reserved

The NETWORK ADDRESS LENGTH field contains the length in bytes of the NETWORK ADDRESS field. The network address length shall be a multiple of four.

The null-terminated, null-padded NETWORK ADDRESS field contains the URL form of a URI as defined in RFC 2396.

### 7.7.6 Mode Page Policy VPD page

The Mode Page Policy VPD page (see table 483) indicates which mode page policy (see 6.9) is in effect for each mode page supported by the logical unit.

**Table 483 — Mode Page Policy VPD page**

Bit Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (87h)							
2	(MSB)							
3	PAGE LENGTH (n-3)							(LSB)
	Mode page policy descriptor list							
4								
7	Mode page policy descriptor [first]							
	⋮							
n-3								
n	Mode page policy descriptor [last]							

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field are as defined in 6.4.2.

The PAGE LENGTH field indicates the length of the mode page policy descriptor list. The relationship between the PAGE LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

Each mode page policy descriptor (see table 484) contains information describing the mode page policy for one or more mode pages or subpages (see 7.4.5). The information in the mode page policy descriptors in this VPD page shall describe the mode page policy for every mode page and subpage supported by the logical unit.

**Table 484 — Mode page policy descriptor**

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved		POLICY PAGE CODE					
1	POLICY SUBPAGE CODE							
2	MLUS	Reserved					MODE PAGE POLICY	
3	Reserved							

The POLICY PAGE CODE field and POLICY SUBPAGE CODE field indicate the mode page and subpage to which the descriptor applies.

If the first mode page policy descriptor in the list contains a POLICY PAGE CODE field set to 3Fh and a POLICY SUBPAGE CODE field set to FFh, then the descriptor applies to all mode pages and subpages not described by other mode page policy descriptors. The POLICY PAGE CODE field shall be set to 3Fh and the POLICY SUBPAGE CODE field shall be set to FFh only in the first mode page policy descriptor in the list.

If the POLICY PAGE CODE field contains a value other than 3Fh and a POLICY SUBPAGE CODE field contains a value other than FFh, then the POLICY PAGE CODE field and the POLICY SUBPAGE CODE field indicate a single mode page and subpage to which the descriptor applies.

If the POLICY PAGE CODE field contains a value other than 3Fh, then POLICY SUBPAGE CODE field shall contain a value other than FFh. If the POLICY SUBPAGE CODE field contains a value other than FFh, then POLICY PAGE CODE field shall contain a value other than 3Fh.

If the SCSI target device has more than one logical unit, a multiple logical units share (MLUS) bit set to one indicates the mode page and subpage identified by the POLICY PAGE CODE field and POLICY SUBPAGE CODE field is shared by more than one logical unit. A MLUS bit set to zero indicates the logical unit maintains its own copy of the mode page and subpage identified by the POLICY PAGE CODE field and POLICY SUBPAGE CODE field.

The MLUS bit is set to one in the mode page policy descriptors or descriptor that indicates the mode page policy for the:

- a) Disconnect-Reconnect mode page (see 7.4.8); and
- b) Protocol Specific Port mode page (see 7.4.14).

The MODE PAGE POLICY field (see table 485) indicates the mode page policy for the mode page and subpage identified by the POLICY PAGE CODE field and POLICY SUBPAGE CODE field. The mode page policies are described in table 151 (see 6.9).

**Table 485 — MODE PAGE POLICY field**

Code	Description
00b	Shared
01b	Per target port
10b	Obsolete
11b	Per I_T nexus

### 7.7.7 SCSI Ports VPD page

The SCSI Ports VPD page (see table 486) provides a means to retrieve designation descriptors for all the SCSI ports in a SCSI target device or SCSI target/initiator device.

**Table 486 — SCSI Ports VPD page**

Bit Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (88h)							
2	(MSB)							
3	PAGE LENGTH (n-3)							
	(LSB)							
	Designation descriptor list							
4	SCSI port designation descriptor [first]							
	(see table 487)							
	⋮							
	SCSI port designation descriptor [last]							
n	(see table 487)							

The SCSI Ports VPD page only reports information on SCSI ports known to the device server processing the INQUIRY command. The REPORT LUNS well-known logical unit (see 8.2) may be used to return information on all SCSI ports in the SCSI device (i.e., all target ports and all initiator ports).

If the device server detects that a SCSI port is added or removed from the SCSI device and the SCSI port designation descriptor list changes, it shall establish a unit attention condition (see SAM-4) for the initiator port associated with every I\_T nexus, with the additional sense code set to INQUIRY DATA HAS CHANGED.

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field are as defined in 6.4.2.

The PAGE LENGTH field indicates the length of the SCSI port designation descriptor list. The relationship between the PAGE LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

Each SCSI port designation descriptor (see table 487) identifies a SCSI port. The SCSI port designation descriptors may be returned in any order.

**Table 487 — SCSI port designation descriptor**

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved							
1								
2	(MSB)	RELATIVE PORT IDENTIFIER						(LSB)
3								
4	Reserved							
5								
6	(MSB)	INITIATOR PORT TRANSPORTID LENGTH (k-7)						(LSB)
7								
8	INITIATOR PORT TRANSPORTID, if any							
k								
k+1	Reserved							
k+2								
k+3	(MSB)	TARGET PORT DESCRIPTORS LENGTH (n-(k+4))						(LSB)
k+4								
Target port descriptor list								
k+5	Target port descriptor [first] (see table 489)							
	⋮							
	Target port descriptor [last] (see table 489)							
n								

The RELATIVE PORT IDENTIFIER field (see table 488) contains the relative port identifier (see 3.1.120) of the SCSI port to which the SCSI port designation descriptor applies.

**Table 488 — RELATIVE PORT IDENTIFIER field**

Code	Description
0h	Reserved
1h	Relative port 1, historically known as port A
2h	Relative port 2, historically known as port B
3h to FFFFh	Relative port 3 through 65 535

The INITIATOR PORT TRANSPORTID LENGTH field contains the length of the INITIATOR PORT TRANSPORTID field. An INITIATOR PORT TRANSPORTID LENGTH field set to zero indicates no INITIATOR PORT TRANSPORTID field is present (i.e., the SCSI port is not an initiator port and not a target/initiator port).

If the INITIATOR PORT TRANSPORTID LENGTH field contains a non-zero value, the INITIATOR PORT TRANSPORTID field contains a TransportID identifying the initiator port as specified in 7.5.4.

The TARGET PORT DESCRIPTORS LENGTH field contains the length of the target port descriptors, if any. A TARGET PORT DESCRIPTORS LENGTH field set to zero indicates no target port descriptors are present (i.e., the SCSI port is not a target port and not a target/initiator port).

Each target port descriptor (see table 489) contains an identifier for the target port. The target port descriptors may be returned in any order.

**Table 489 — Target port descriptor**

Bit Byte	7	6	5	4	3	2	1	0
0	PROTOCOL IDENTIFIER				CODE SET			
1	PIV (1b)	Reserved	ASSOCIATION (01b)		DESIGNATOR TYPE			
2	Reserved							
3	DESIGNATOR LENGTH (n-3)							
4	DESIGNATOR							
n								

The PROTOCOL IDENTIFIER field indicates the SCSI transport protocol to which the designation descriptor applies as described in 7.7.3.1.

The CODE SET field, PIV field, ASSOCIATION field, DESIGNATOR TYPE field, DESIGNATOR LENGTH field, and DESIGNATOR field are as defined in the Device Identification VPD page designation descriptor (see 7.7.3.1), with the following additional requirements:

- a) The PIV bit shall be set to one (i.e., the PROTOCOL IDENTIFIER field always contains a SCSI transport protocol identifier); and
- b) The ASSOCIATION field shall be set to 01b (i.e., the descriptor always identifies a target port).

### 7.7.8 Protocol Specific Logical Unit Information VPD page

The Protocol Specific Logical Unit Information VPD page (see table 490) contains protocol specific parameters associated with a SCSI port that may be different for each logical unit in the SCSI target device.

**Table 490 — Protocol Specific Logical Unit Information VPD page**

Bit Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (90h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Logical unit information descriptor list							
4	Logical unit information descriptor [first]							
	(see table 491)							
	⋮							
	Logical unit information descriptor [last]							
n	(see table 491)							

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field are as defined in 6.4.2.

The PAGE LENGTH field indicates the length of the logical unit information descriptor list. The relationship between the PAGE LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The logical unit information descriptor list contains descriptors for SCSI ports known to the device server that is processing the INQUIRY command. The logical unit information descriptors (see table 491) may be returned in any order.

**Table 491 — Logical unit information descriptor**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____							
1	RELATIVE PORT IDENTIFIER _____ (LSB)							
2	Reserved				PROTOCOL IDENTIFIER			
3	_____							
5	Reserved _____							
6	(MSB) _____							
7	PROTOCOL SPECIFIC DATA LENGTH (n-7) _____ (LSB)							
8	_____							
n	Per logical unit SCSI transport protocol specific data _____							

The RELATIVE PORT IDENTIFIER field contains the relative port identifier (see 3.1.120) of the SCSI port to which the logical unit information descriptor applies and is defined in the SCSI Ports VPD page (see 7.7.7).

The PROTOCOL IDENTIFIER field indicates the SCSI transport protocol to which the logical unit information descriptor applies as described in 7.7.3.1.

The PROTOCOL SPECIFIC DATA LENGTH field indicates the length in bytes of the per logical unit SCSI transport protocol specific data.

The per logical unit SCSI transport protocol specific data is defined by the SCSI transport protocol standard (see 3.1.140) that corresponds to the SCSI target port.

### 7.7.9 Protocol Specific Port Information VPD page

The Protocol Specific Port Information VPD page (see table 492) contains protocol specific parameters associated with a SCSI port that are the same for all logical units in the SCSI target device.

**Table 492 — Protocol Specific Port Information VPD page**

Bit Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (91h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Port information descriptor list							
4	Port information descriptor [first] (see table 491)							
	⋮							
n	Port information descriptor [last] (see table 491)							

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field are as defined in 6.4.2.

The PAGE LENGTH field indicates the length of the port information descriptor list. The relationship between the PAGE LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.



The port information descriptor list contains descriptors for SCSI ports known to the device server that is processing the INQUIRY command. The port information descriptors (see table 493) may be returned in any order.

**Table 493 — Port information descriptor**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)	RELATIVE PORT IDENTIFIER						(LSB)
1								
2	Reserved				PROTOCOL IDENTIFIER			
3								
5	Reserved							
6	(MSB)	PROTOCOL SPECIFIC DATA LENGTH (n-7)						(LSB)
7								
8								
n	Shared SCSI transport protocol specific data							

The RELATIVE PORT IDENTIFIER field contains the relative port identifier (see 3.1.120) of the SCSI port to which the port information descriptor applies and is defined in the SCSI Ports VPD page (see 7.7.7).

The PROTOCOL IDENTIFIER field indicates the SCSI transport protocol to which the port information descriptor applies as described in 7.7.3.1.

The PROTOCOL SPECIFIC DATA LENGTH field indicates the length in bytes of the shared SCSI transport protocol specific data.

The shared SCSI transport protocol specific data is defined by the SCSI transport protocol standard (see 3.1.140) that corresponds to the SCSI target port.

### 7.7.10 Software Interface Identification VPD page

The Software Interface Identification VPD page (see table 494) provides identification of software interfaces applicable to the logical unit. Logical units may have more than one associated software interface identifier.

NOTE 79 - Application clients may use the software IDs to differentiate peripheral device function in cases where the command set (e.g., processor devices) is too generic to distinguish different software interfaces implemented.

**Table 494 — Software Interface Identification VPD page**

Bit Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (84h)							
2	Reserved							
3	PAGE LENGTH (n-3)							
	Software interface identifier list							
4	Software interface identifier [first]							
	⋮							
n	Software interface identifier [last]							

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field are defined in 6.4.2.

The PAGE LENGTH field indicates the length of the software interface identifier list. The relationship between the PAGE LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

Each software interface identifier (see table 495) is a six-byte, fixed-length field that contains information identifying a software interface implemented by the logical unit. The contents of software interface identifier are in EUI-48 format (see 3.1.51).

**Table 495 — Software interface identifier format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
2	IEEE COMPANY_ID							(LSB)
3	(MSB)							
5	VENDOR SPECIFIC EXTENSION IDENTIFIER							(LSB)

The IEEE COMPANY\_ID field contains a 24 bit OUI (see 3.1.97) assigned by the IEEE.

The VENDOR SPECIFIC EXTENSION IDENTIFIER a 24 bit numeric value that is uniquely assigned by the organization associated with the OUI as required by the IEEE definition of EUI-48 (see 3.1.51). The combination of OUI and vendor specific extension identifier shall uniquely identify the document or documents that specify the supported software interface.

### 7.7.11 Supported VPD Pages VPD page

The Supported VPD Pages VPD page contains a list of the VPD page codes supported by the logical unit (see table 496). If a device server supports any VPD pages, it also shall support this VPD page.

**Table 496 — Supported VPD Pages VPD page**

Bit Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (00h)							
2	Reserved							
3	PAGE LENGTH (n-3)							
4	Supported VPD page list							
n								

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field are defined in 6.4.2.

The PAGE LENGTH field indicates the length of the supported VPD page list. The relationship between the PAGE LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The supported VPD page list shall contain a list of all VPD page codes (see 7.7) implemented by the logical unit in ascending order beginning with page code 00h.

### 7.7.12 Unit Serial Number VPD page

The Unit Serial Number VPD page (see table 497) provides a product serial number for the SCSI target device or logical unit.

**Table 497 — Unit Serial Number VPD page**

Bit Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (80h)							
2	Reserved							
3	PAGE LENGTH (n-3)							
4	(MSB)							
n	PRODUCT SERIAL NUMBER							
	(LSB)							

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field are defined in 6.4.2.

The PAGE LENGTH field indicates the length of the product serial number. The relationship between the PAGE LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The PRODUCT SERIAL NUMBER field contains right-aligned ASCII data (see 4.4.1) that is vendor-assigned serial number. If the product serial number is not available, the device server shall return ASCII spaces (20h) in this field.

## 8 Well known logical units

### 8.1 Model for well known logical units

Well known logical units are addressed using the well known logical unit addressing method of extended logical unit addressing (see SAM-4). Each well known logical unit has a well known logical unit number (W-LUN) as shown in table 498.

**Table 498 — Well known logical unit numbers**

W-LUN	Description	Reference
00h	Reserved	
01h	REPORT LUNS well known logical unit	8.2
02h	ACCESS CONTROLS well known logical unit	8.3
03h	TARGET LOG PAGES well known logical unit	8.4
04h	SECURITY PROTOCOL well known logical unit	8.5
05h	MANAGEMENT PROTOCOL well known logical unit	8.6
06h-FFh	Reserved	

If a well known logical unit is supported within a SCSI target device, then that logical unit shall support all the commands defined for it.

Access to well known logical units shall not be affected by access controls.

The SCSI device name of the well known logical unit may be determined by issuing an INQUIRY command (see 6.4) requesting the Device Identification VPD page (see 7.7.3).

All well known logical units shall support the INQUIRY command's Device Identification VPD page as specified in 7.7.3.2.2.

### 8.2 REPORT LUNS well known logical unit

The REPORT LUNS well known logical unit shall only process the commands listed in table 499. If a command is received by the REPORT LUNS well known logical unit that is not listed in table 499, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID COMMAND OPERATION CODE.

**Table 499 — Commands for the REPORT LUNS well known logical unit**

Command name	Operation code	Type	Reference
INQUIRY	12h	M	6.4
REPORT LUNS	A0h	M	6.23
REQUEST SENSE	03h	M	6.29
TEST UNIT READY	00h	M	6.37
Key: M = Command implementation is mandatory.			

## 8.3 ACCESS CONTROLS well known logical unit

### 8.3.1 Access controls model

#### 8.3.1.1 Access controls commands

The ACCESS CONTROLS well known logical unit shall only process the commands listed in table 500. If a command is received by the ACCESS CONTROLS well know logical unit that is not listed in table 500, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID COMMAND OPERATION CODE.

**Table 500 — Commands for the ACCESS CONTROLS well known logical unit**

Command name	Operation code	Type	Reference
ACCESS CONTROL IN	86h	M	8.3.2
ACCESS CONTROL OUT	87h	M	8.3.3
INQUIRY	12h	M	6.4
REQUEST SENSE	03h	M	6.29
TEST UNIT READY	00h	M	6.37
Key: M = Command implementation is mandatory.			

#### 8.3.1.2 Access controls overview

Access controls are a SCSI target device feature that application clients may use to restrict logical unit access to specified initiator ports or groups of initiator ports.

Access controls shall not allow restrictions to be placed on access to well known logical units. Access controls shall not cause new well known logical units to be defined.

Access controls are handled in the SCSI target device by an access controls coordinator located at the ACCESS CONTROLS well known logical unit. The access controls coordinator also may be accessible via LUN 0. The access controls coordinator associates a specific LUN to a specific logical unit depending on which initiator port accesses the SCSI target device and whether the initiator port has access rights to the logical unit.

Access rights to a logical unit affects whether the logical unit appears in the parameter data returned by a REPORT LUNS command and how the logical unit responds to INQUIRY commands.

The access controls coordinator maintains the ACL as described in 8.3.1.3 to supply information about:

- Which initiator ports are allowed access to which logical units; and
- Which LUN value is used by a specific initiator port when accessing a specific logical unit.

The format of the ACL is vendor specific.

To support third party commands (e.g., EXTENDED COPY), the access controls coordinator may provide proxy tokens (see 8.3.1.6.2) to allow an application client to pass its access capabilities to the application client for another initiator port.

An application client manages the access controls state of the SCSI target device using the ACCESS CONTROL IN command (see 8.3.2) and the ACCESS CONTROL OUT command (see 8.3.3).

A SCSI target device has access controls disabled when it is manufactured and after successful completion of the ACCESS CONTROL OUT command with DISABLE ACCESS CONTROLS service action (see 8.3.3.3). When access controls are disabled, the ACL contains no entries and the management identifier key (see 8.3.1.8) is zero.

The first successful ACCESS CONTROL OUT command with MANAGE ACL service action (see 8.3.3.2) shall enable access controls. When access controls are enabled, all logical units, except LUN 0 and all well known logical units, shall be inaccessible to all initiator ports unless the ACL (see 8.3.1.3) allows access.

The ACL allows an initiator port access to a logical unit if the ACL contains an ACE (see 8.3.1.3) with an access identifier (see 8.3.1.3.2) associated with the initiator port and that ACE contains a LUACD (see 8.3.1.3.3) that references the logical unit.

When the ACL allows access to a logical unit, the REPORT LUNS command parameter data bytes representing that logical unit shall contain the LUN value found in the LUACD that references that logical unit and the application client for the initiator port shall use the same LUN value when sending commands to the logical unit.

An initiator port also may be allowed access to a logical unit through the use of a proxy token (see 8.3.1.6.2).

Once access controls are enabled, they shall remain enabled until:

- a) Successful completion of an ACCESS CONTROL OUT command with DISABLE ACCESS CONTROLS service action; or
- b) Vendor specific physical intervention.

Successful downloading of microcode (see 6.39) may result in access controls being disabled.

Once access controls are enabled, power cycles, hard resets, logical unit resets, and I\_T nexus losses shall not disable them.

### **8.3.1.3 The access control list (ACL)**

#### **8.3.1.3.1 ACL overview**

The specific access controls for a SCSI target device are instantiated by the access controls coordinator using data in an ACL. The ACL contains zero or more ACEs and zero or more proxy tokens (see 8.3.1.6.2.1).

Each ACE contains the following:

- a) One access identifier (see 8.3.1.3.2) that identifies the initiator port(s) to which the ACE applies; and
- b) A list of LUACDs (see 8.3.1.3.3) that identify the logical units to which the identified initiator port(s) have access rights and the LUNs used to access those logical units via those initiator port(s). Each LUACD contains the following:
  - A) A vendor specific logical unit reference; and
  - B) A LUN value.

Figure 16 shows the logical structure of an ACL.

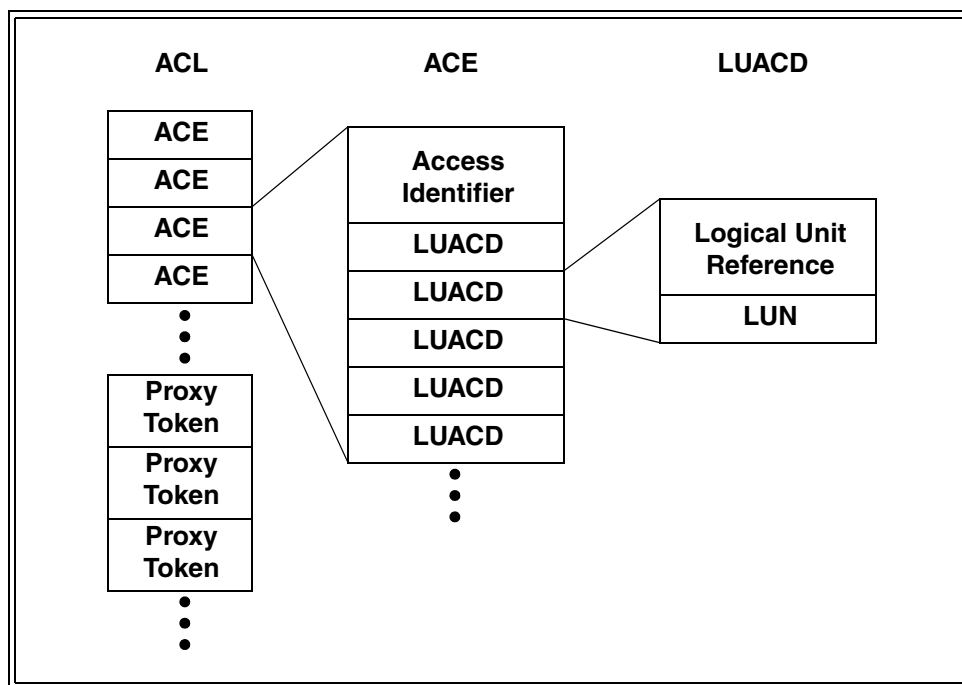


Figure 16 — ACL Structure

#### 8.3.1.3.2 Access identifiers

Initiator ports are identified in an ACE using one of the following types of access identifiers:

- a) AccessID - based on initiator port enrollment;
- b) TransportID - based on protocol specific identification of initiator ports; or
- c) Vendor specific access identifiers.

An initiator port is allowed access to the logical units in an ACE containing an AccessID type access identifier when that initiator port is enrolled as described in 8.3.1.5. An initiator port that has not previously enrolled uses the ACCESS CONTROL OUT command with ACCESS ID ENROLL service action to enroll including the AccessID in parameter data as specified in 8.3.3.4.

An initiator port is associated with an AccessID type access identifier if that initiator port is in the enrolled state or pending-enrolled state with respect to that AccessID (see 8.3.1.5). At any instant in time, an initiator port may be associated with at most one AccessID. All initiator ports enrolled using a specific AccessID share the same ACE and access to all the logical units its LUACDs describe.

TransportID access identifiers are SCSI transport protocol specific as described in 7.5.4.

An initiator port is allowed access to the logical units in an ACE containing a TransportID type access identifier when the identification for the initiator port matches that found in the TransportID in a way that is consistent with the TransportID definition (see 7.5.4). There is no need to process any command to obtain logical unit access based on a Transport ID because the needed information is provided by the SCSI transport protocol layer.

The formats of access identifiers are defined in 8.3.1.13.

### 8.3.1.3.3 Logical unit access control descriptors

Each LUACD in an ACE identifies one logical unit to which the initiator ports associated with the access identifier are allowed access and specifies the LUN value used when accessing the logical unit via those initiator ports. The format of a LUACD is vendor specific.

The identification of a logical unit in a LUACD is vendor specific. The logical unit identified by a LUACD shall not be a well known logical unit. A logical unit shall be referenced in no more than one LUACD per ACE.

The LUN value shall conform to the requirements specified in SAM-4. A specific LUN value shall appear in no more than one LUACD per ACE.

### 8.3.1.4 Managing the ACL

#### 8.3.1.4.1 ACL management overview

The contents of the ACL are managed by an application client using the ACCESS CONTROL OUT command with MANAGE ACL and DISABLE ACCESS CONTROLS service actions. The ACCESS CONTROL OUT command with MANAGE ACL service action (see 8.3.3.2) is used to add, remove, or modify ACEs thus adding, revoking, or changing the allowed access of initiator ports to logical units. The ACCESS CONTROL OUT command with DISABLE ACCESS CONTROLS service action (see 8.3.3.3) disables access controls and discards the ACL.

#### 8.3.1.4.2 Authorizing ACL management

To reduce the possibility of applications other than authorized ACL managers changing the ACL, successful completion of specific access controls service actions (e.g., ACCESS CONTROL OUT command with MANAGE ACL or DISABLE ACCESS CONTROLS service action) requires delivery of the correct management identifier key value (see 8.3.1.8) in the ACCESS CONTROL OUT parameter data. The service actions that require the correct management identifier key are shown in table 501 and table 502.

**Table 501 — ACCESS CONTROL OUT management identifier key requirements**

Service action name	Management Identifier Key Required	Reference
ACCESS ID ENROLL	No	8.3.3.4
ASSIGN PROXY LUN	No	8.3.3.11
CANCEL ENROLLMENT	No	8.3.3.5
CLEAR ACCESS CONTROLS LOG	Yes	8.3.3.6
DISABLE ACCESS CONTROLS	Yes	8.3.3.3
MANAGE ACL	Yes	8.3.3.2
MANAGE OVERRIDE LOCKOUT TIMER	Yes/No	8.3.3.7
OVERRIDE MGMT ID KEY	No	8.3.3.8
RELEASE PROXY LUN	No	8.3.3.12
REVOKE ALL PROXY TOKENS	No	8.3.3.10
REVOKE PROXY TOKEN	No	8.3.3.9



**Table 502 — ACCESS CONTROL IN management identifier key requirements**

<b>Service action name</b>	<b>Management Identifier Key Required</b>	<b>Reference</b>
REPORT ACCESS CONTROLS LOG	Yes	8.3.2.4
REPORT ACL	Yes	8.3.2.2
REPORT LU DESCRIPTORS	Yes	8.3.2.3
REPORT OVERRIDE LOCKOUT TIMER	Yes	8.3.2.5
REQUEST PROXY TOKEN	No	8.3.2.6

#### **8.3.1.4.3 Identifying logical units during ACL management**

The access controls coordinator shall identify every logical unit of a SCSI target device with a unique default LUN value. The default LUN values used by the access controls coordinator shall be the LUN values that would be reported by the REPORTS LUNS command if access controls were disabled.

An application client discovers the default LUN values using the ACCESS CONTROL IN command with REPORT LU DESCRIPTORS (see 8.3.2.3) or REPORT ACL (see 8.3.2.2) service action and then supplies those default LUN values to the access controls coordinator using the ACCESS CONTROL OUT command with MANAGE ACL service action.

The association between default LUN values and logical units is managed by the access controls coordinator and may change due to circumstances that are beyond the scope of this standard. To track such changes, the access controls coordinator maintains the DLgeneration counter as described in 8.3.1.4.4.

#### **8.3.1.4.4 Tracking changes in logical unit identification**

The access controls coordinator shall implement a Default LUNs Generation (DLgeneration) counter to track changes in the association between default LUN values and logical units. The DLgeneration counter is a wrapping counter.

When access controls are disabled the DLgeneration counter shall be set to zero. When access controls are first enabled (see 8.3.1.2) the DLgeneration counter shall be set to one. While access controls are enabled, the access controls coordinator shall increment the DLgeneration counter by one every time the association between default LUN values and logical units changes (e.g., following the creation of a new logical unit, deletion of an existing logical unit, or removal and recreation of an existing logical unit).

The access controls coordinator shall include the current DLgeneration value in the parameter data returned by an ACCESS CONTROL IN command with REPORT LU DESCRIPTORS (see 8.3.2.3) or REPORT ACL (see 8.3.2.2) service action. The application client shall supply the DLgeneration value for the default LUN values it is using in the parameter data for an ACCESS CONTROL OUT command with MANAGE ACL service action (see 8.3.3.2).

Before processing the ACL change information in the parameter list provided by an ACCESS CONTROL OUT command with MANAGE ACL service action, the access controls coordinator shall verify that the DLgeneration value in the parameter data matches the current value of the DLgeneration counter. If the DLgeneration value verification finds a mismatch, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

### 8.3.1.5 Enrolling AccessIDs

#### 8.3.1.5.1 Enrollment states

##### 8.3.1.5.1.1 Summary of enrollment states

Application clients enroll an initiator port AccessID with the access controls coordinator to be allowed to access logical units listed in the ACE having the same AccessID type access identifier. The ACCESS CONTROL OUT command with ACCESS ID ENROLL service action (see 8.3.3.4) is used to enroll an AccessID. An initiator port shall be in one of three states with respect to such an enrollment:

- a) **Not-enrolled:** The state for an initiator port before the first ACCESS CONTROL OUT command with ACCESS ID ENROLL service action is sent to the access controls coordinator. Also the state for an initiator port following successful completion of an ACCESS CONTROL OUT command with CANCEL ENROLLMENT service action (see 8.3.3.5);
- b) **Enrolled:** The state for an initiator port following successful completion of an ACCESS CONTROL OUT command with ACCESS ID ENROLL service action; or
- c) **Pending-enrolled:** The state for an enrolled initiator port following:
  - A) Events described in 8.3.1.12; or
  - B) Successful completion of an ACCESS CONTROL OUT command with MANAGE ACL service action from any initiator port with the FLUSH bit set to one (see 8.3.3.2).

##### 8.3.1.5.1.2 Not-enrolled state

The access controls coordinator shall place an initiator port in the not-enrolled state when it first detects the receipt of a SCSI command or task management function from that initiator port. The initiator port shall remain in the not-enrolled state until successful completion of an ACCESS CONTROL OUT command with ACCESS ID ENROLL service action (see 8.3.3.5).

When in the not-enrolled state, an initiator port shall only have access to logical units on the basis of a TransportID (see 8.3.1.3.2) or on the basis of proxy tokens (see 8.3.1.6.2.1).

The access controls coordinator changes an initiator port from the enrolled state or pending-enrolled state to the not-enrolled state in response to the following events:

- a) Successful completion of the ACCESS CONTROL OUT command with CANCEL ENROLLMENT service action (see 8.3.3.5) shall change the state to not-enrolled; or
- b) Successful completion of an ACCESS CONTROL OUT command with MANAGE ACL service action (see 8.3.3.2) that replaces the ACL entry for the enrolled AccessID as follows:
  - A) If the NOCNCL bit (see 8.3.3.2.2) is set to zero in the ACCESS CONTROL OUT command with MANAGE ACL service action parameter data, the state shall change to not-enrolled; or
  - B) If the NOCNCL bit is set to one, the state may change to not-enrolled based on vendor specific criteria.

An application client for an enrolled initiator port may discover that the initiator port has transitioned to the not-enrolled state as a result of actions taken by a third party (e.g., an ACCESS CONTROL OUT command with MANAGE ACL service action performed by another initiator port or a logical unit reset).

Placing an enrolled initiator in the not-enrolled state indicates that the ACE defining that initiator port's logical unit access has changed (e.g., previous relationships between logical units and LUN values may no longer apply).

If an application client detects this loss of enrollment on an initiator port, it may take recovery actions. However, such actions may be disruptive for the SCSI initiator device and may not be required. Use of the not-enrolled state is avoidable if the application client that sends the ACCESS CONTROL OUT command with MANAGE ACL service

action determines that its requested changes to the ACL do not alter the existing relationships between logical units and LUN values in any existing ACEs with AccessID type access identifiers and sets the NOCNCL bit to one, recommending that initiator ports be left in their current enrollment state.

The access controls coordinator selects from the following options for responding to a NOCNCL bit set to one in a vendor specific manner:

- a) Honor the recommendation, causing the minimum effects on SCSI initiator devices and requiring no extra actions on the part of the access controls coordinator;
- b) Ignore the recommendation and always place initiator ports in the non-enrolled state, causing the maximum disruption for SCSI initiator devices, but requiring no extra resources on the part of the access controls coordinator; or
- c) Ignore the recommendation and examine the current and new ACEs to determine if an initiator port should be placed in the non-enrolled state.

If the application client that sends the ACCESS CONTROL OUT command with MANAGE ACL service action is unable to determine whether the ACE logical unit relationships are altered as a result of processing the command, then it should set the NOCNCL bit to zero and it should coordinate the ACL change with the application clients for affected initiator ports to ensure proper data integrity. Such coordination is beyond the scope of this standard.

#### 8.3.1.5.1.3 Enrolled state

The access controls coordinator shall place an initiator port in the enrolled state (i.e., enroll the initiator port) following successful completion of the ACCESS CONTROL OUT command with ACCESS ID ENROLL service action (see 8.3.3.4). The ACCESS CONTROL OUT command with ACCESS ID ENROLL service action is successful only if:

- a) The initiator port was in the not-enrolled state and the AccessID in the ACCESS CONTROL OUT command with ACCESS ID ENROLL service action parameter data matches the access identifier in an ACE. This results in the initiator port being enrolled and allowed access to the logical units specified in the LUACDs in the ACE (see 8.3.1.3); or
- b) The initiator port was in the enrolled state or pending-enrolled state and the AccessID in the ACCESS CONTROL OUT command with ACCESS ID ENROLL service action parameter data matches the current enrolled AccessID for the initiator port.

If the initiator port was in the enrolled state or pending-enrolled state and the AccessID in the ACCESS CONTROL OUT command with ACCESS ID ENROLL service action parameter data does not match the current enrolled AccessID for the initiator port, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - ENROLLMENT CONFLICT. If the initiator port was in the enrolled state, it shall be transitioned to the pending-enrolled state.

Transitions from the enrolled state to the not-enrolled state are described in 8.3.1.5.1.2. Transitions from the enrolled state to the pending-enrolled state are described in 8.3.1.5.1.4.

NOTE 80 - This standard does not preclude implicit enrollments through mechanisms in a service delivery subsystem. Such mechanisms should perform implicit enrollments after identification by TransportID and should fail in the case where there are ACL conflicts as described in 8.3.1.5.2.

#### 8.3.1.5.1.4 Pending-enrolled state

The access controls coordinator shall place an initiator port in the pending-enrolled state if that initiator port currently is in the enrolled state, and in response to the following:

- a) A logical unit reset;
- b) An I\_T nexus loss associated with that initiator port; or
- c) Successful completion of an ACCESS CONTROL OUT command with MANAGE ACL service action where the FLUSH bit is set to one in the parameter data.

While in the pending-enrolled state, the initiator port's access to logical units is limited as described in 8.3.1.7.

#### 8.3.1.5.2 ACL LUN conflict resolution

ACL LUN conflicts may occur if:

- a) An application client for an initiator port in the not-enrolled state attempts to enroll an AccessID using the ACCESS CONTROL OUT command with ACCESS ID ENROLL service action (see 8.3.3.4); or
- b) An ACCESS CONTROL OUT command with MANAGE ACL service action (see 8.3.3.2) attempts to change the ACL with the result that it conflicts with existing enrollments (see 8.3.1.5) or proxy LUN assignments (see 8.3.1.6.2.2).

Three types of ACL LUN conflicts may occur:

- a) The TransportID ACE (see 8.3.1.3) and the AccessID ACE for the initiator port each contain a LUACD with the same LUN value but with different logical unit references;
- b) The TransportID ACE and the AccessID ACE for the initiator port each contain a LUACD with the different LUN values but with the same logical unit references; or
- c) The enrolling initiator port has proxy access rights to a logical unit addressed with a LUN value that equals a LUN value in a LUACD in the AccessID ACE for the initiator port.

If an ACL LUN conflict occurs during the processing of an ACCESS CONTROL OUT command with MANAGE ACL service action, the command shall be terminated with CHECK CONDITION status (see 8.3.3.2.2).

If an ACL LUN conflict occurs during the processing of an ACCESS CONTROL OUT command with ACCESS ID ENROLL service action, the following actions shall be taken as part of the handling of the enrollment function:

- a) The ACCESS CONTROL OUT command with ACCESS ID ENROLL service action shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, the additional sense code set to ACCESS DENIED - ACL LUN CONFLICT;
- b) The initiator port shall remain in the not-enrolled state; and
- c) If the ACL LUN conflict is not the result of proxy access rights, the access controls coordinator shall record the event in the access controls log as described in 8.3.1.10.

#### 8.3.1.6 Granting and revoking access rights

##### 8.3.1.6.1 Non-proxy access rights

The ACCESS CONTROL OUT command with MANAGE ACL service action (see 8.3.3.2) adds or replaces ACEs in the ACL (see 8.3.1.3). One ACE describes the logical unit access allowed by one access identifier (see 8.3.1.3.2) and the LUN values to be used in addressing the accessible logical units. The access identifier specifies the initiator port(s) permitted access to the logical units described by the ACE.

With the exception of proxy access rights (see 8.3.1.6.2), access rights are granted by:

- a) Adding a new ACE to the ACL; or
- b) Replacing an existing ACE with an ACE that includes additional LUACDs.

With the exception of proxy access rights, access rights are revoked by:

- a) Removing an ACE from the ACL; or
- b) Replacing an existing ACE with an ACE that removes one or more LUACDs.

When an ACE is added or replaced the requirements stated in 8.3.1.5.1.2 and 8.3.1.11 apply.

### **8.3.1.6.2 Proxy access**

#### **8.3.1.6.2.1 Proxy tokens**

An application client with access rights to a logical unit via an initiator port on the basis of an ACE in the ACL (see 8.3.1.6.1) may temporarily share that access with third parties using the proxy access mechanism. The application client uses the ACCESS CONTROL IN command with REQUEST PROXY TOKEN service action (see 8.3.2.6) to request that the access control coordinator generate a proxy token for the logical unit specified by the LUN value in the CDB.

The access controls coordinator generates the proxy token in a vendor specific manner. For a specific SCSI target device, all active proxy token values should be unique. Proxy token values should be reused as infrequently as possible to prevent proxy tokens that have been used and released from being given unintended meaning.

Power cycles, hard resets, logical unit resets, and I\_T nexus losses shall not affect the validity and proxy access rights of proxy tokens (see 8.3.1.12). A proxy token shall remain valid and retain the same proxy access rights until one of the following occurs:

- a) An application client with access rights to a logical unit via an initiator port based on an ACE in the ACL revokes the proxy token using:
  - A) The ACCESS CONTROL OUT command with REVOKE PROXY TOKEN service action (see 8.3.3.9); or
  - B) The ACCESS CONTROL OUT command with REVOKE ALL PROXY TOKENS service action (see 8.3.3.10); or
- b) An application client issues the ACCESS CONTROL OUT command with MANAGE ACL service action (see 8.3.3.2) with parameter data containing the Revoke Proxy Token ACE page (see 8.3.3.2.4) or Revoke All Proxy Tokens ACE page (see 8.3.3.2.5).

#### **8.3.1.6.2.2 Proxy LUNs**

To extend proxy access rights to a third party, an application client forwards a proxy token (see 8.3.1.6.2.2) to the third party (e.g., in a target descriptor in the parameter data of the EXTENDED COPY command).

The third party sends the access controls coordinator an ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action (see 8.3.3.11) specifying the proxy token to request creation of a proxy access right to the referenced logical unit. The access controls coordinator determines the referenced logical unit from the proxy token value. The third party is unaware of the exact logical unit to which it is requesting access.

The parameter data for the ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action includes the LUN value that the third party intends to use when accessing the referenced logical unit. The resulting LUN value is called a proxy LUN. If the ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action is successful, the proxy LUN becomes the third party's mechanism for accessing the logical unit by proxy.

Once assigned, a proxy LUN shall remain valid until one of the following occurs:

- a) The third party releases the proxy LUN value using the ACCESS CONTROL OUT command with RELEASE PROXY LUN service action (see 8.3.3.12);
- b) The proxy token is made invalid as described in 8.3.1.6.2.1; or
- c) A logical unit reset or I\_T nexus loss of the I\_T nexus used to assign the proxy LUN (see 8.3.1.12).

The third party may reissue the ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action in an attempt to re-establish its proxy access rights. If the cause of the proxy token becoming invalid was temporary, the reissued command should succeed. The access controls coordinator shall process the request as described in 8.3.1.6.2.1 without reference to any previous assignment of the proxy LUN value.

### 8.3.1.7 Verifying access rights

When access controls are enabled (see 8.3.1.2), access rights for an initiator port shall be validated as described in this subclause.

Relationships between access controls and tasks in a task set are described in 8.3.1.11.1.

All commands shall be processed as if access controls were not present if the ACL (see 8.3.1.3) allows the initiator port access to the addressed logical unit as a result of one of the following conditions:

- a) The ACL contains an ACE containing a TransportID type access identifier (see 8.3.1.3.2) for the initiator port and that ACE includes a LUACD with LUN value matching the addressed LUN;
- b) The initiator port is in the enrolled state (see 8.3.1.5.1.3) under an AccessID, the ACL contains an ACE containing that AccessID as an access identifier, and that ACE includes a LUACD with LUN value matching the addressed LUN; or
- c) The addressed LUN matches a proxy LUN value (see 8.3.1.6.2.2) assigned using the ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action (see 8.3.3.11) and the proxy token (see 8.3.1.6.2.1) used to assign the proxy LUN value is still valid.

If the initiator port is in the pending-enrolled state (see 8.3.1.5.1.4) under an AccessID, the ACL contains an ACE containing that AccessID as an access identifier, and that ACE includes a LUACD with LUN value matching the addressed LUN, then commands shall be processed as follows:

- a) INQUIRY, REPORT LUNS, ACCESS CONTROL OUT and ACCESS CONTROL IN commands shall be processed as if access controls were not present;
- b) A REQUEST SENSE command (see 6.29) shall be processed as if access controls were not present, except in cases where parameter data containing sense data with a sense key set to NO SENSE would be returned (see 6.29 and table 356 in 7.4.11). In these cases, device server shall terminate the REQUEST SENSE command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST and the additional sense code set to ACCESS DENIED - INITIATOR PENDING-ENROLLED; and
- c) Any other command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - INITIATOR PENDING-ENROLLED.

An application client should respond to the ACCESS DENIED - INITIATOR PENDING-ENROLLED additional sense code by sending an ACCESS CONTROL OUT command with ACCESS ID ENROLL service action. If the command succeeds, the application client may retry the terminated command.

If an INQUIRY command is addressed to a LUN for which there is no matching LUN value in any LUACD in any ACE allowing the initiator port logical unit access rights, the standard INQUIRY data (see 6.4.2) PERIPHERAL DEVICE TYPE field shall be set to 1Fh and the PERIPHERAL QUALIFIER field shall be set to 011b (i.e., the device server is not capable of supporting a device at this logical unit).

The parameter data returned in response to a REPORT LUNS command addressed to LUN 0 or to the REPORT LUNS well known logical unit shall return only the list of LUN values that are associated to accessible logical units according to the following criteria:

- a) If the initiator port is in the enrolled state or pending-enrolled state, the REPORT LUNS parameter data shall include any LUN values found in LUACDs in the ACE containing the AccessID enrolled by the initiator port;
- b) If the initiator port, in any enrollment state has a TransportID found in the access identifier of an ACE, then the REPORT LUNS parameter data shall include any LUN values found in LUACDs in that ACE; and
- c) If the initiator port, in any enrollment state has access to any proxy LUNs (see 8.3.1.6.2.2), then those LUN values shall be included in the REPORT LUNS parameter data.

The parameter data returned in response to a REPORT LUNS command that describes well known logical units shall not be affected by access controls.

If the initiator port is in the not-enrolled state and is not allowed access to any logical unit as result of its TransportID or as a result of a proxy LUN assignment, then the REPORT LUNS parameter data shall include only LUN 0 and well known logical units, as specified in 6.23.

Except when access controls are disabled, all cases not described previously in this subclause shall result in termination of the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to LOGICAL UNIT NOT SUPPORTED.

### **8.3.1.8 The management identifier key**

#### **8.3.1.8.1 Management identifier key usage**

The management identifier key identifies the application that is responsible for managing access controls for a SCSI target device. This identification occurs when the application client specifies a new management identifier key value in each ACCESS CONTROL OUT command with the MANAGE ACL service action (see 8.3.3.2), and when the last specified management identifier key value appears in ACCESS CONTROL IN and ACCESS CONTROL OUT service actions as required in 8.3.1.4.2.

To allow for failure scenarios where the management identifier key value has been lost, an override procedure involving a timer is described in 8.3.1.8.2.

Use of the management identifier key has the following features:

- a) Management of access controls is associated with those application clients that provide the correct management identifier key without regard for the initiator port from which the command was received; and
- b) Only an application client that has knowledge of the management identifier key may change the ACL, allowing the management of access controls to be limited to specific applications and application clients.

### **8.3.1.8.2 Overriding the management identifier key**

#### **8.3.1.8.2.1 The OVERRIDE MGMT ID KEY service action**

If the management identifier key needs to be replaced and the current management identifier key is not available, then the ACCESS CONTROL OUT command with OVERRIDE MGMT ID KEY service action (see 8.3.3.8) may be used to force the management identifier key to a known value.

The ACCESS CONTROL OUT command with OVERRIDE MGMT ID KEY service action should be used only for failure recovery. If failure recovery is not required, the ACCESS CONTROL OUT command with MANAGE ACL service action should be used.

To protect the management identifier key from unauthorized overrides, the access controls coordinator shall restrict use of the ACCESS CONTROL OUT command with OVERRIDE MGMT ID KEY service action based on the value of the override lockout timer (see 8.3.1.8.2.2).

When the override lockout timer is not zero, an ACCESS CONTROL OUT command with OVERRIDE MGMT ID KEY service action shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

When the override lockout timer is zero, an ACCESS CONTROL OUT command with OVERRIDE MGMT ID KEY service action shall be processed as described in 8.3.3.8.

The access controls coordinator shall log the receipt of each ACCESS CONTROL OUT command with OVERRIDE MGMT ID KEY service action and its success or failure as described in 8.3.1.10.

#### **8.3.1.8.2.2 The override lockout timer**

The access controls coordinator shall maintain the override lockout timer capable of counting up to 65 535 seconds. When the override lockout timer is not zero it shall be decreased by one nominally once per second but no more frequently than once every 800 milliseconds until the value reaches zero. When the override lockout timer is zero, it shall not be changed except as the result of commands sent by an application client.

The ACCESS CONTROL OUT command with MANAGE OVERRIDE LOCKOUT TIMER service action manages the state of the override lockout timer (see 8.3.3.7), performing one of the following functions:

- a) If the incorrect management identifier key is supplied or if no parameter data is sent, the access controls coordinator shall reset the override lockout timer to the last received initial override lockout timer value; or
- b) If the correct management identifier key is supplied, then the access controls coordinator shall do the following:
  - 1) Save the initial override lockout timer value supplied in the parameter data; and
  - 2) Reset the override lockout timer to the new initial value.

Setting the initial override lockout timer value to zero disables the override lockout timer and allows the ACCESS CONTROL OUT command with OVERRIDE MGMT KEY service action to succeed at any time.

Any application that knows the management identifier key may establish an initial override lockout timer value of sufficient duration (i.e., up to about 18 hours). Maintaining a non-zero override lockout timer value may be accomplished without knowing the management identifier key or transporting the management identifier key on a service delivery subsystem. Attempts to establish a zero initial override lockout timer value that are not accompanied by the correct management identifier key result in decreasing the probability that a subsequent ACCESS CONTROL OUT command with OVERRIDE MGMT ID KEY service action is able to succeed by resetting the override lockout timer.



After a logical unit reset, the override lock timer shall be set to the initial override lockout timer value within ten seconds of the non-volatile memory containing the initial override lockout timer value becoming available.

The ACCESS CONTROL IN command with REPORT OVERRIDE LOCKOUT TIMER may be used to discover the state of the override lockout timer.

#### 8.3.1.9 Reporting access control information

Specific service actions of the ACCESS CONTROL IN command may be used by an application client to request a report from the access controls coordinator about its access controls data and state.

The ACCESS CONTROL IN command with REPORT ACL service action (see 8.3.2.2) returns the ACL (see 8.3.1.3). The information reported includes the following:

- a) The list of access identifiers (see 8.3.1.3.2) and the associated LUACDs (see 8.3.1.3.3) currently in effect; and
- b) The list of proxy tokens (see 8.3.1.6.2.1) currently in effect.

The ACCESS CONTROL IN command with REPORT ACCESS CONTROLS LOG service action (see 8.3.2.4) returns the contents of the access controls log (see 8.3.1.10).

The ACCESS CONTROL IN command with REPORT OVERRIDE LOCKOUT TIMER service action (see 8.3.2.5) reports on the state of the override lockout timer (see 8.3.1.8.2.2).

#### 8.3.1.10 Access controls log

The access controls log is a record of events maintained by the access controls coordinator.

The access controls log has three portions, each recording a different class of events:

- a) **invalid key events**: mismatches between the management identifier key (see 8.3.1.8) specified by a service action and the current value maintained by the access controls coordinator;
- b) **key override events**: attempts to override the management identifier key (see 8.3.1.8.2.1), whether the attempt fails or succeeds; and
- c) **ACL LUN conflict events**: (see 8.3.1.5.2).

Each portion of the log is required to contain a saturating counter of the corresponding events. When a SCSI target device is manufactured, all event counters shall be set to zero. When access controls are disabled, all event counters except the Key Override events counter shall be set to zero. Each event counter shall be incremented by one whenever the corresponding event occurs.

Each log portion may contain additional records with more specific information about each event. When the resources for additional log records are exhausted, the access controls coordinator shall preserve the most recently added log records in preference to older log records.

Log records contain a TIME STAMP field whose contents are vendor specific. If the access controls coordinator has no time stamp resources the TIME STAMP field shall be set to zero. If time stamp values are provided, the same timing clock and time stamp format shall be used for all access controls log entries.

Invalid key events occur whenever an access controls command requires the checking of an application client supplied management identifier key against the current management identifier key saved by the access controls coordinator and the two values fail to match. When such an event occurs, the access controls coordinator shall

increment the Invalid Keys events counter by one. If the log has additional resources to record event details, the access controls coordinator shall add an invalid keys log record (containing the information defined in 8.3.2.4.2.3) describing the event.

Key override events occur when the access controls coordinator receives the ACCESS CONTROL OUT command with OVERRIDE MGMT KEY service action (see 8.3.3.8). When such an event occurs, the access controls coordinator shall increment the Key Overrides events counter by one without regard for whether the command succeeds or fails. If the log has additional resources to record event details, the access controls coordinator shall add a key overrides log record (containing the information defined in 8.3.2.4.2.2) describing the event.

ACL LUN conflict events occur as specified in 8.3.1.5.2. When such an event occurs, the access controls coordinator shall increment the ACL LUN Conflicts events counter by one. If the log has additional resources to record event details, the access controls coordinator shall add an ACL LUN conflicts log record (containing the information defined in 8.3.2.4.2.4) describing the event.

Selected portions of the access controls log may be requested by an application client using the ACCESS CONTROL IN command with REPORT ACCESS CONTROLS LOG service action (see 8.3.2.4). With the exception of the key overrides portion, selected portions of the log may be cleared and the event counters reset to zero using the ACCESS CONTROL OUT command with CLEAR ACCESS CONTROLS LOG service action (see 8.3.3.6).

### **8.3.1.11 Interactions of access controls and other features**

#### **8.3.1.11.1 Task set management and access controls**

Upon successful completion of an ACCESS CONTROL OUT command with MANAGE ACL service action (see 8.3.3.2), the specified ACL (see 8.3.1.3) shall apply to all tasks that subsequently enter the task enabled state. Tasks that have modified SCSI target device state information (e.g., media, mode pages, and log pages) shall not be affected by an ACCESS CONTROL OUT command that subsequently enters the task enabled state. Tasks in the task enabled state that have not modified SCSI target device state information may or may not be affected by an ACCESS CONTROL OUT command that subsequently enters the task enabled state. The ACL in effect prior to when the ACCESS CONTROL OUT command with MANAGE ACL or DISABLE ACCESS CONTROLS service action entered the task enabled state shall apply to all tasks that are not affected by the ACCESS CONTROL OUT command.

All the operations performed by a task shall complete under the control of a single ACL, either the state in effect prior to processing of the ACCESS CONTROL OUT command or the state in effect following processing of the ACCESS CONTROL OUT command. After a task enters the task enabled state for the first time changing the access control state from disabled to enabled (see 8.3.1.2) shall have no effect on the task.

Multiple access control commands, both ACCESS CONTROL IN and ACCESS CONTROL OUT, may be in the task set concurrently. The order of processing of such commands is defined by the task set management requirements (see SAM-4), but each command shall be processed as a single indivisible command without any interleaving of actions that may be required by other access control commands.

#### **8.3.1.11.2 Existing reservations and ACL changes**

If a logical unit is reserved by one I\_T nexus and that logical unit becomes accessible to another I\_T nexus as a result of an access control command, then there shall be no changes in the reservation of that logical unit.

If a logical unit is reserved by an I\_T nexus and that logical unit becomes inaccessible to that I\_T nexus as a result of an access control command or other access control related event, then there shall be no changes in the reservation. Existing persistent reservations mechanisms allow for other SCSI initiator devices with access to that logical unit to clear the reservation.

### 8.3.1.12 Access controls information persistence and memory usage requirements

If a SCSI target device supports access controls, then the SCSI target device shall contain an access controls coordinator that shall maintain the following information in nonvolatile memory:

- a) Whether access controls are enabled or disabled; and
- b) The access controls data that table 503 and table 504 require to persist across power cycles, hard resets, and logical unit resets.

If the access control coordinator's nonvolatile memory is not ready and the access controls coordinator is unable to determine that access controls are disabled, then the device servers for all logical units shall terminate all commands except INQUIRY and REQUEST SENSE commands with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set as described in table 270 (see 6.37). If a device server in any logical unit receives an INQUIRY command or a REQUEST SENSE command while the access control coordinator's nonvolatile memory is not ready and the access controls coordinator is unable to determine that access controls are disabled, then the device server shall process the command.

Following an I\_T nexus loss, a previously enrolled initiator port shall be placed in the pending-enrolled state, if that initiator port was associated with the lost I\_T nexus. Following a logical unit reset, all previously enrolled initiator ports shall be placed in the pending-enrolled state.

The information shown in table 503 shall be maintained by the access controls coordinator.

**Table 503 — Mandatory access controls resources**

Information Description	Size (in bits)	Persistent Across Power Cycles, Hard Resets, and Logical Unit Resets
One ACL (see 8.3.1.3) containing at least one ACE containing one access identifier (see 8.3.1.3.2), and at least one LUACD (see 8.3.1.3.3)	VS	Yes
The Enrollment State for each initiator port (see 8.3.1.5.1)	VS	Yes
Management Identifier Key (see 8.3.1.8)	64	Yes
Default LUNs Generation (DLgeneration, see 8.3.1.4.4)	32	Yes
Override Lockout Timer (see 8.3.1.8.2.2)	16	No
Initial Override Lockout Timer value (see 8.3.1.8.2.2)	16	Yes
Access Controls Log Event Counters (see 8.3.1.10) containing at least the following:		Yes
a) Key Override events counter;	16	Yes
b) Invalid Key events counter; and	16	Yes
c) ACL LUN Conflict events counter	16	Yes

Optionally, the access controls coordinator may maintain the information shown in table 504.

**Table 504 — Optional access controls resources**

Information Description	Size (in bits)	Persistent Across Power Cycles, Hard Resets, and Logical Unit Resets
One or more proxy tokens (see 8.3.1.6.2.1)	64	Yes
One or more proxy LUNs (see 8.3.1.6.2.2)	64	No
Access controls log event records (see 8.3.1.10) for:		
a) Key Override events;	(see 8.3.2.4.2.2)	Yes
b) Invalid Key events; and	(see 8.3.2.4.2.3)	Yes
c) ACL LUN Conflict events	(see 8.3.2.4.2.4)	Yes

At the time of manufacturer, the ACL shall be empty, all values shown in table 503 shall be zero, additional access control log structures shall be empty and there shall be no valid proxy tokens.

### 8.3.1.13 Access identifier formats

#### 8.3.1.13.1 Access identifier type

The ACCESS IDENTIFIER TYPE field (see table 505) indicates the format and usage of the access identifier.

**Table 505 — ACCESS IDENTIFIER TYPE field**

Code	Access Identifier Name	Reference
00h	AccessID	8.3.1.13.2
01h	TransportID	7.5.4
02h to 7Fh	Reserved	
80h to FFh	Vendor specific	

#### 8.3.1.13.2 AccessID access identifiers

AccessID access identifiers shall have the format shown in table 506.

**Table 506 — AccessID access identifier format**

Bit Byte	7	6	5	4	3	2	1	0
0	ACCESSID							
15								
16	Reserved							
23								

The ACCESSID field contains a value that identifies the AccessID type ACE in which the AccessID access identifier appears. Within the ACL, no two ACEs shall contain the same AccessID.

### 8.3.2 ACCESS CONTROL IN command

#### 8.3.2.1 ACCESS CONTROL IN introduction

The service actions of the ACCESS CONTROL IN command (see table 507) are used to obtain information about the access controls that are active within the access controls coordinator and to perform other access control functions (see 8.3.1). If the ACCESS CONTROL IN command is implemented, the ACCESS CONTROL OUT command also shall be implemented. The ACCESS CONTROL IN command shall not be affected by access controls.

**Table 507 — ACCESS CONTROL IN service actions**

Service Action	Name	Type	Reference
00h	REPORT ACL	M	8.3.2.2
01h	REPORT LU DESCRIPTORS	M	8.3.2.3
02h	REPORT ACCESS CONTROLS LOG	M	8.3.2.4
03h	REPORT OVERRIDE LOCKOUT TIMER	M	8.3.2.5
04h	REQUEST PROXY TOKEN	O	8.3.2.6
05h to 17h	Reserved		
18h to 1Fh	Vendor specific		
Key: M = Service action implementation is mandatory if ACCESS CONTROL IN is implemented. O = Service action implementation is optional.			

The ACCESS CONTROL IN command may be addressed to any logical unit whose standard INQUIRY data (see 6.4.2) has the ACC bit set to one (e.g., LUN 0), in which case it shall be processed in the same manner as if the command had been addressed to the ACCESS CONTROLS well known logical unit. If an ACCESS CONTROL IN command is received by a device server whose standard INQUIRY data has the ACC bit set to zero, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID COMMAND OPERATION CODE.

### 8.3.2.2 REPORT ACL service action

#### 8.3.2.2.1 REPORT ACL introduction

The ACCESS CONTROL IN command with REPORT ACL service action (see table 508) is used to query the ACL (see 8.3.1.3) maintained by the access controls coordinator. If the ACCESS CONTROL IN command is implemented, the REPORT ACL service action shall be implemented.

**Table 508 — ACCESS CONTROL IN command with REPORT ACL service action**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (86h)							
1	Reserved			SERVICE ACTION (00h)				
2	(MSB) _____							
9	MANAGEMENT IDENTIFIER KEY _____ (LSB)							
10	(MSB) _____							
13	ALLOCATION LENGTH _____ (LSB)							
14	Reserved							
15	CONTROL							

If access controls are disabled, the device server shall ignore the MANAGEMENT IDENTIFIER KEY field and shall complete the command with GOOD status returning the eight byte parameter list header specified in 8.3.2.2.2 subject to the allocation length limitation described in 4.3.5.6.

If access controls are enabled and the contents of the MANAGEMENT IDENTIFIER KEY field do not match the current management identifier key (see 8.3.1.8) maintained by the access controls coordinator, then parameter data shall not be returned and the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - INVALID MGMT ID KEY. This event shall be recorded in the invalid keys portion of the access controls log (see 8.3.1.10).

The ALLOCATION LENGTH field is defined in 4.3.5.6. The ALLOCATION LENGTH field value should be at least eight.

### 8.3.2.2.2 REPORT ACL parameter data format

#### 8.3.2.2.2.1 REPORT ACL parameter data introduction

The format of the parameter data returned in response to an ACCESS CONTROL IN command with REPORT ACL service actions is shown in table 509.

**Table 509 — ACCESS CONTROL IN with REPORT ACL parameter data format**

Bit Byte	7	6	5	4	3	2	1	0
	Parameter list header							
0	(MSB)	ACL DATA LENGTH (n-3)						(LSB)
3								
4	(MSB)	DLGENERATION						(LSB)
7								
	ACL data pages							
8		ACL data page 0						
		⋮						
n		ACL data page x						

The ACL DATA LENGTH field shall contain a count of the number of bytes in the remaining parameter data. If access controls are disabled, the ACL DATA LENGTH field shall be set to four. The relationship between the ACL DATA LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The DLGENERATION field shall contain the current DLgeneration value (see 8.3.1.4.4).

The ACL data pages contain a description of the ACL (see 8.3.1.3) maintained by the access controls coordinator. Each ACL data page describes one ACE in the ACL or one proxy token (see 8.3.1.6.2). Every ACE and every proxy token managed by the access controls coordinator shall have an ACL data page in the parameter data. The content and format of an ACL data page is indicated by a page code. Table 510 lists the ACL data page codes.

**Table 510 — ACL data page codes**

Page Code	ACL Data Page Name	Reference
00h	Granted	8.3.2.2.2.2
01h	Granted All	8.3.2.2.2.3
02h	Proxy Tokens	8.3.2.2.2.4
03h to EFh	Reserved	
F0h to FFh	Vendor specific	

### 8.3.2.2.2.2 Granted ACL data page format

The Granted ACL data page (see table 511) describes an ACE that allows access to a specific set of logical units via a list of LUACDs (see 8.3.1.3.3).

**Table 511 — Granted ACL data page format**

Bit Byte	7	6	5	4	3	2	1	0
0	PAGE CODE (00h)							
1	Reserved							
2	(MSB)	PAGE LENGTH (n-3)						(LSB)
3								
4	Reserved							
5	ACCESS IDENTIFIER TYPE							
6	(MSB)	ACCESS IDENTIFIER LENGTH (m-7)						(LSB)
7								
8	ACCESS IDENTIFIER							
m								
	LUACD Descriptors							
m+1	LUACD descriptor 0							
m+20								
	⋮							
n-19	LUACD descriptor x							
n								

The PAGE LENGTH field indicates the number of additional bytes required for this ACL data page. The relationship between the PAGE LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The ACCESS IDENTIFIER TYPE field (see 8.3.1.13) indicates the format and usage of the access identifier.

The ACCESS IDENTIFIER LENGTH field indicates the number of bytes following taken up by the ACCESS IDENTIFIER field. The access identifier length shall be at least 24 and shall be a multiple of four.

The ACCESS IDENTIFIER field contains the identifier that the access controls coordinator uses to select the initiator port(s) that are allowed access to the logical units named by the LUACD descriptors in this ACL data page. The format of the ACCESS IDENTIFIER field is specified in table 505 (see 8.3.1.13). One Granted or Granted All (see 8.3.2.2.2.3) ACL data page shall be returned for a specific pair of values in the ACCESS IDENTIFIER TYPE and ACCESS IDENTIFIER fields.



Each LUACD descriptor (see table 512) describes the access allowed to one logical unit based on the access identifier. There shall be one LUACD descriptor for each logical unit to which the access identifier allows access.

**Table 512 — Granted ACL data page LUACD descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	ACCESS MODE							
1	Reserved							
3								
4	LUN VALUE							
11								
12	DEFAULT LUN							
19								

The ACCESS MODE field (see table 513) indicates the type of access allowed to the logical unit referenced by the DEFAULT LUN field and addressable at the specified LUN value.

**Table 513 — ACCESS MODE field**

Code	Description
00h	Normal access
01h to EFh	Reserved
F0h to FFh	Vendor specific

The LUN VALUE field indicates the LUN value an accessing application client uses to access the logical unit via the initiator port to which the LUACD descriptor applies.

The DEFAULT LUN field identifies the logical unit to which access is allowed using the default LUN value described in 8.3.1.4.3. The value in the DEFAULT LUN field shall be consistent with the DLGENERATION field value returned in the parameter list header (see 8.3.2.2.2).

The LUN VALUE and DEFAULT LUN fields may contain the same value.

### 8.3.2.2.2.3 Granted All ACL data page format

The Granted All ACL data page (see table 514) describes an ACE that allows access to all the SCSI target device's logical units with the default LUN values being used as the accessing LUN values. Initiator ports that have access via the access identifier in a Granted All ACL data page are allowed to access the SCSI target device as if access controls were disabled.

**Table 514 — Granted All ACL data page format**

Bit Byte	7	6	5	4	3	2	1	0
0	PAGE CODE (01h)							
1	Reserved							
2	(MSB)	PAGE LENGTH (m-3)						(LSB)
3								
4	Reserved							
5	ACCESS IDENTIFIER TYPE							
6	(MSB)	ACCESS IDENTIFIER LENGTH (m-7)						(LSB)
7								
8	ACCESS IDENTIFIER							
m								

The PAGE LENGTH, ACCESS IDENTIFIER TYPE, and ACCESS IDENTIFIER LENGTH, are described in 8.3.2.2.2.2.

The ACCESS IDENTIFIER field contains the identifier that the access controls coordinator uses to select the initiator port(s) that are allowed access to all the SCSI target device's logical units with the default LUN values being used as the accessing LUN values. The format of the access identifier field is specified in table 505 (see 8.3.1.13). One Granted (see 8.3.2.2.2.2) or Granted All ACL data page shall be returned for a specific pair of values in the ACCESS IDENTIFIER TYPE and ACCESS IDENTIFIER fields.

### 8.3.2.2.2.4 Proxy Tokens ACL data page format

The Proxy Tokens ACL data page (see table 515) describes the proxy tokens (see 8.3.1.6.2) maintained by the access controls coordinator.

**Table 515 — Proxy Tokens ACL data page format**

Bit Byte	7	6	5	4	3	2	1	0
0	PAGE CODE (02h)							
1	Reserved							
2	(MSB)	PAGE LENGTH (n-3)						(LSB)
3								
	Proxy token descriptors							
4	Proxy token descriptor 0							
23								
	⋮							
n-19	Proxy token descriptor x							
n								

The PAGE LENGTH field indicates the number of additional bytes required for this ACL data page. The relationship between the PAGE LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

If there are no active proxy tokens, the access controls coordinator may either not include the Proxy Tokens ACL data page in the parameter data or may include one such ACL data page containing no proxy token descriptors.

No more than one Proxy Tokens ACL data page shall be included in the parameter data.

Each proxy token descriptor (see table 516) describes the access allowed to one logical unit based on one proxy token. There shall be one proxy token descriptor for each active proxy token maintained by the access controls coordinator.

**Table 516 — Proxy token descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved							
3								
4								
11	PROXY TOKEN							
12	DEFAULT LUN							
19								

The PROXY TOKEN field indicates the proxy token to which this proxy token descriptor applies.

The DEFAULT LUN field identifies the logical unit to which this proxy token allows access using the default LUN value described in 8.3.1.4.3. The value in the DEFAULT LUN field shall be consistent with the DLGENERATION field value returned in the parameter list header (see 8.3.2.2.2).

The same default LUN value may appear in multiple proxy token descriptors, if multiple proxy tokens are valid for the same logical unit.

8.3.2.3 REPORT LU DESCRIPTORS service action

8.3.2.3.1 REPORT LU DESCRIPTORS introduction

The ACCESS CONTROL IN command with REPORT LU DESCRIPTORS service action (see table 517) reports the inventory of logical units for which access controls may be established. If the ACCESS CONTROL IN command is implemented, the REPORT LU DESCRIPTORS service action shall be implemented.

Table 517 — ACCESS CONTROL IN command with REPORT LU DESCRIPTORS service action

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (86h)							
1	Reserved			SERVICE ACTION (01h)				
2	(MSB) <div>MANAGEMENT IDENTIFIER KEY</div> (LSB)							
9	(MSB) <div>ALLOCATION LENGTH</div> (LSB)							
10	Reserved							
13	CONTROL							

If access controls are disabled, the device server shall ignore the MANAGEMENT IDENTIFIER KEY field and shall complete the command with GOOD status returning the 20 byte parameter list header as specified in 8.3.2.3.2 subject to the ALLOCATION LENGTH limitation described in 4.3.5.6.

NOTE 81 - When access controls are disabled, the logical unit inventory may be obtained using commands such as REPORT LUNS (see 6.23). To facilitate access controls management, the ACCESS CONTROL IN command with REPORT LU DESCRIPTORS service action returns more information than the REPORT LUNS command. When access controls are disabled additional commands such as INQUIRY (see 6.4) are required to obtain all the information provided by the ACCESS CONTROL IN command with REPORT LU DESCRIPTORS service action.

If access controls are enabled and the contents of the MANAGEMENT IDENTIFIER KEY field do not match the current management identifier key (see 8.3.1.8) maintained by the access controls coordinator, then parameter data shall not be returned and the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - INVALID MGMT ID KEY. This event shall be recorded in the invalid keys portion of the access controls log (see 8.3.1.10).

The ALLOCATION LENGTH field is defined in 4.3.5.6. The ALLOCATION LENGTH field value should be at least 20.

### 8.3.2.3.2 REPORT LU DESCRIPTORS parameter data format

The format of the parameter data returned in response to an ACCESS CONTROL IN command with REPORT LU DESCRIPTORS service actions is shown in table 518.

**Table 518 — ACCESS CONTROL IN with REPORT LU DESCRIPTORS parameter data format**

Bit Byte	7	6	5	4	3	2	1	0
	Parameter list header							
0	(MSB)	LU INVENTORY LENGTH (n-3)						(LSB)
3								
4	(MSB)	NUMBER OF LOGICAL UNITS						(LSB)
7								
8		SUPPORTED LUN MASK FORMAT						
15								
16	(MSB)	DLGENERATION						(LSB)
19								
	Logical Unit descriptors							
20		Logical Unit descriptor 0						
		⋮						
n		Logical Unit descriptor x						

The LU INVENTORY LENGTH field shall contain a count of the number of bytes in the remaining parameter data. If access controls are disabled, the LU INVENTORY LENGTH field shall be set to 16. The relationship between the LU INVENTORY LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The NUMBER OF LOGICAL UNITS field shall contain a count of the number of logical units managed by the access controls coordinator. The value in NUMBER OF LOGICAL UNITS field shall be the same as the number of Logical Unit descriptors that follow in the parameter data.

The SUPPORTED LUN MASK FORMAT field (see table 519) contains a summary of the LUN values (see 8.3.1.3.3) that the access controls coordinator supports. LUN values are exchanged between application clients and the access controls coordinator by several service actions (e.g., the ACCESS CONTROL IN command with REPORT ACL service action described in 8.3.2.2 and the ACCESS CONTROL OUT command with MANAGE ACL service action described in 8.3.3.2). The format of the SUPPORTED LUN MASK FORMAT field follows the eight byte LUN structure defined for dependent logical units by SAM-4.

**Table 519 — SUPPORTED LUN MASK FORMAT field format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)	FIRST LEVEL LUN MASK						(LSB)
1								
2	(MSB)	SECOND LEVEL LUN MASK						(LSB)
3								
4	(MSB)	THIRD LEVEL LUN MASK						(LSB)
5								
6	(MSB)	FOURTH LEVEL LUN MASK						(LSB)
7								

The LUN MASK at each level indicates the approximate range of the logical unit number values the access controls coordinator supports. A bit set to zero in a LUN MASK field indicates that the access controls coordinator prohibits setting that bit to one in a LUN value. A bit set to one in a LUN MASK field indicates that the access controls coordinator may allow setting that bit to one in a LUN value.

(E.g., if the access controls coordinator only supports level one LUN values with LUN values ranging from 0 to 256, then the SUPPORTED LUN MASK FORMAT field shall contain 00FF 0000 0000 0000h. If only LUN values ranging from 0 to 200 were supported, the SUPPORTED LUN MASK FORMAT field still would contain 00FF 0000 0000 0000h.)

The value in the SUPPORT LUN MASK FORMAT field only summarizes the supported LUN values and is not a complete description. The value in the SUPPORT LUN MASK FORMAT field should be used as a guideline for specifying LUN values in service actions (e.g., ACCESS CONTROL OUT command with MANAGE ACL service action). LUN values that appear valid based on the contents of the SUPPORT LUN MASK FORMAT field may still be rejected.

The DLGENERATION field shall contain the current DLgeneration value (see 8.3.1.4.4).

Each Logical Unit descriptor (see table 520) contains information about one logical unit managed by the access controls coordinator. There shall be one Logical Unit descriptor for every logical unit managed by the access controls coordinator.

**Table 520 — Logical Unit descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved			PERIPHERAL DEVICE TYPE				
1	Reserved							
2	(MSB)							
3	DESCRIPTOR LENGTH (n-3)							(LSB)
4	DEFAULT LUN							
11	Reserved							
12	EVPD DESIGNATION DESCRIPTOR LENGTH							
13	Reserved							
14	DEVICE IDENTIFIER LENGTH							
15	EVPD DESIGNATION DESCRIPTOR							
16	EVPD DESIGNATION DESCRIPTOR							
47	EVPD DESIGNATION DESCRIPTOR							
48	(MSB)							
79	DEVICE IDENTIFIER							(LSB)
80	DEVICE TYPE SPECIFIC DATA							
n	DEVICE TYPE SPECIFIC DATA							

The PERIPHERAL DEVICE TYPE field is as defined in 6.4.2.

The DESCRIPTOR LENGTH field indicates the total number of bytes remaining in the descriptor. If the PERIPHERAL DEVICE TYPE field contains 0h, 4h, or 7h, the DESCRIPTOR LENGTH field shall contain 92 if the descriptor includes the DEVICE TYPE SPECIFIC DATA field and 80 if it does not. If the PERIPHERAL DEVICE TYPE field contains any value other than 0h, 4h, or 7h, the DESCRIPTOR LENGTH field shall contain 76. The relationship between the DESCRIPTOR LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The DEFAULT LUN field contains the default LUN value (see 8.3.1.4.3) for the logical unit described by this logical unit descriptor. The value in the DEFAULT LUN field shall be consistent with the DLGENERATION field value returned in the parameter list header (see 8.3.2.3.2). The value in the DEFAULT LUN field shall not identify a well known logical unit.

The EVPD DESIGNATION DESCRIPTOR LENGTH field indicates the number of non pad bytes in the EVPD DESIGNATION DESCRIPTOR field.

The DEVICE IDENTIFIER LENGTH field indicates the number of non pad bytes in the DEVICE IDENTIFIER field.

The EVPD DESIGNATION DESCRIPTOR field shall be derived from one of the Device Identification VPD page (see 7.7.3) designation descriptors having 00b in the ASSOCIATION field as follows:

- a) If the designation descriptor has a length less than 32 bytes, then the EVPD DESIGNATION DESCRIPTOR field shall be set to the zero-padded (see 4.4.2) designation descriptor value. The EVPD DESIGNATION DESCRIPTOR LENGTH field shall be set to the length of the designation descriptor not including pad bytes; or
- b) If the designation descriptor has a length greater than or equal to 32 bytes, then the EVPD DESIGNATION DESCRIPTOR field shall be set to the first 32 bytes of the designation descriptor. The EVPD DESIGNATION DESCRIPTOR LENGTH field shall be set to 32.

If there are several designation descriptors having 00b in the ASSOCIATION field, the choice of which descriptor to copy to the EVPD DESIGNATION DESCRIPTOR field is vendor specific, however, all ACCESS CONTROL IN commands with REPORT LU DESCRIPTORS service action shall return the same EVPD DESIGNATION DESCRIPTOR field contents for a specific logical unit.

If a device identifier has been set for the logical unit using the SET DEVICE IDENTIFIER command (see 6.33), the DEVICE IDENTIFIER field shall contain that device identifier subject to the following considerations:

- a) If the device identifier has length less than 32 bytes, then the DEVICE IDENTIFIER field shall be set to the zero-padded (see 4.4.2) device identifier value. The DEVICE IDENTIFIER LENGTH field shall be set to the length of the device identifier not including pad bytes; or
- b) If the device identifier has length greater than or equal to 32 bytes, then the DEVICE IDENTIFIER field shall be set to the first 32 bytes of the identifier. The DEVICE IDENTIFIER LENGTH field shall be set to 32.

If no device identifier has been established by a SET DEVICE IDENTIFIER command, then the DEVICE IDENTIFIER LENGTH field shall be set to zero and the DEVICE IDENTIFIER field shall be set to zero.

If the PERIPHERAL DEVICE TYPE field contains any value other than 0h, 4h, or 7h, the DEVICE TYPE SPECIFIC DATA field shall not be present in the Logical Unit descriptor.

The Logical Unit descriptor shall include the DEVICE TYPE SPECIFIC DATA field if:

- a) The PERIPHERAL DEVICE TYPE field contains 0h, 4h, or 7h;
- b) The logical unit supports the READ CAPACITY command (see SBC-2) with:
  - A) The RELADR bit set to zero; and
  - B) The PMI bit set to zero;and
- c) The logical unit standard INQUIRY data (see 6.4.2) has the RMB bit set to zero.

If the Logical Unit descriptor includes the DEVICE TYPE SPECIFIC DATA field, then the size of the DEVICE TYPE SPECIFIC DATA field shall be 12 bytes and the field shall contain data equivalent to that returned by a successful READ CAPACITY command with LONGLBA bit set to one, and the RELADR and PMI bits set to zero.



### 8.3.2.4 REPORT ACCESS CONTROLS LOG service action

#### 8.3.2.4.1 REPORT ACCESS CONTROLS LOG introduction

The ACCESS CONTROL IN command with REPORT ACCESS CONTROLS LOG service action (see table 521) is used to obtain the access controls log (see 8.3.1.10). If the ACCESS CONTROL IN command is implemented, the REPORT ACCESS CONTROLS LOG service action shall be implemented.

**Table 521 — ACCESS CONTROL IN command with REPORT ACCESS CONTROLS LOG service action**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (86h)							
1	Reserved			SERVICE ACTION (02h)				
2	(MSB)							
9	MANAGEMENT IDENTIFIER KEY (LSB)							
10	Reserved					LOG PORTION		
11	Reserved							
12	(MSB)							
13	ALLOCATION LENGTH (LSB)							
14	Reserved							
15	CONTROL							

If access controls are disabled, the device server shall ignore the MANAGEMENT IDENTIFIER KEY field and shall complete the command with GOOD status returning the eight byte parameter list header as specified in 8.3.2.4.2.1 subject to the ALLOCATION LENGTH limitation described in 4.3.5.6.

Since the Key Overrides portion of the log is maintained while access controls are disabled (see 8.3.3.3), it may be retrieved by enabling access controls and issuing an ACCESS CONTROL IN command with REPORT ACCESS CONTROLS LOG service action.

If access controls are enabled and table 522 specifies that the management identifier key is not required then the device server shall ignore the contents of the MANAGEMENT IDENTIFIER KEY field.

If access controls are enabled, table 522 specifies that the management key identifier is required, and the contents of the MANAGEMENT IDENTIFIER KEY field do not match the current management identifier key (see 8.3.1.8) maintained by the access controls coordinator, then the parameter data shall not be returned and the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - INVALID MGMT ID KEY. This event shall be recorded in the invalid keys portion of the access controls log (see 8.3.1.10).

The LOG PORTION field (see table 522) specifies the access controls log portion being requested.

**Table 522 — CDB LOG PORTION field**

Code	Description	Management Identifier Key Required
00b	Key Overrides portion	No
01b	Invalid Keys portion	Yes
10b	ACL LUN Conflicts portion	Yes
11b	Reserved	

The ALLOCATION LENGTH field is defined in 4.3.5.6. The ALLOCATION LENGTH field value should be at least eight.

#### 8.3.2.4.2 REPORT ACCESS CONTROLS LOG parameter data format

##### 8.3.2.4.2.1 REPORT ACCESS CONTROLS LOG parameter data introduction

The format of the parameter data returned in response to an ACCESS CONTROL IN command with REPORT ACCESS CONTROLS LOG service actions is shown in table 523.

**Table 523 — ACCESS CONTROL IN with REPORT ACCESS CONTROLS LOG parameter data format**

Bit Byte	7	6	5	4	3	2	1	0
	Parameter list header							
0	(MSB) LOG LIST LENGTH (n-3) (LSB)							
3								
4	Reserved							
5	Reserved						LOG PORTION	
6	(MSB) COUNTER (LSB)							
7								
	Access Controls Log portion pages							
8	Access Controls Log portion page 0							
	⋮							
n	Access Controls Log portion page x							

The LOG LIST LENGTH field shall contain a count of the number of bytes in the remaining parameter data. If access controls are disabled, the LOG LIST LENGTH field shall be set to four. The relationship between the LOG LIST LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.5.6.

The LOG PORTION field (see table 524) indicates the access controls log portion being returned, the contents of the COUNTER field, and the type of access controls log portion pages being returned.

**Table 524 — Parameter data LOG PORTION field**

Code	Access Controls Log Portion Being Returned	COUNTER Field Contents	Access Controls Log Page Format Reference
00b	Key Overrides portion	Key Override events counter	8.3.2.4.2.2
01b	Invalid Keys portion	Invalid Key events counter	8.3.2.4.2.3
11b	ACL LUN Conflicts portion	ACL LUN Conflict events counter	8.3.2.4.2.4
11b	Reserved		

The COUNTER field contains the events counter value (see 8.3.1.10) for the access controls log portion indicated by the LOG PORTION field (see table 524).

The format of the access controls log portion pages is indicated by the value in the LOG PORTION field (see table 524). All the access controls log portion pages returned in a single parameter list shall have the same format. If the access controls coordinator does not support access controls log portion pages in the portion of the access controls log indicated by the LOG PORTION field, then the parameter data shall only contain the parameter list header.

#### 8.3.2.4.2.2 Key Overrides access controls log portion page format

The Key Overrides access controls log portion page (see table 525) contains details of logged attempts to override the management identifier key (see 8.3.1.10) using the ACCESS CONTROL OUT command with OVERRIDE MGMT ID KEY service action (see 8.3.3.8) whether those attempts were successful or not.

**Table 525 — Key Overrides access controls log portion page format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)	TRANSPORTID ADDITIONAL LENGTH (m-32)						(LSB)
1		Reserved						
2		Reserved						
3		SUCCESS						
4	(MSB)	TIME STAMP						(LSB)
7		TRANSPORTID						
8		INITIAL OVERRIDE LOCKOUT TIMER						(LSB)
m-1		OVERRIDE LOCKOUT TIMER						(LSB)
m	(MSB)							
m+1								
m+2	(MSB)							
m+3								

The TRANSPORTID ADDITIONAL LENGTH field indicates the additional length of the TRANSPORTID field beyond the minimum length of 24 bytes. The TransportID additional length shall be a multiple of four.

A SUCCESS bit set to one indicates that the specific ACCESS CONTROL OUT command with OVERRIDE MGMT ID KEY service action event recorded in the access controls log successfully overrode the management identifier key. A SUCCESS bit set to zero indicates that the command did not succeed.

The TIME STAMP field shall contain zero or an indication of the time at which the ACCESS CONTROL OUT command with OVERRIDE MGMT ID KEY service action was processed as described in 8.3.1.10.

The TRANSPORTID field shall contain the TransportID of the initiator port from which the command was received.

The INITIAL OVERRIDE LOCKOUT TIMER field shall contain the access controls coordinator's initial override lockout timer value (see 8.3.1.8.2.2) at the time when the key override event was logged.

The OVERRIDE LOCKOUT TIMER field shall contain the access controls coordinator's override lockout timer value (see 8.3.1.8.2.2) at the time when the key override event was logged.

#### 8.3.2.4.2.3 Invalid Keys access controls log portion page format

The Invalid Keys access controls log portion page (see table 526) contains details of logged receipts of ACCESS CONTROL IN or ACCESS CONTROL OUT commands specifying an incorrect management identifier key (see 8.3.1.10).

**Table 526 — Invalid Keys access controls log portion page format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) _____							
1	TRANSPORTID ADDITIONAL LENGTH (m-32)							(LSB)
16	OPERATION CODE							
17	Reserved			SERVICE ACTION				
18	(MSB) _____							
7	TIME STAMP							(LSB)
8								
m-1	TRANSPORTID							_____
m	(MSB) _____							
m+7	INVALID MANAGEMENT IDENTIFIER KEY							(LSB)

The TRANSPORTID ADDITIONAL LENGTH field indicates the additional length of the TRANSPORTID field beyond the minimum length of 24 bytes. The TransportID additional length shall be a multiple of four.

The OPERATION CODE and SERVICE ACTION fields shall be set to the respective values from the CDB of the access controls command that specified the invalid management identifier key.

The TIME STAMP field shall contain zero or an indication of the time at which the ACCESS CONTROL IN or ACCESS CONTROL OUT command was processed as described in 8.3.1.10.

The TRANSPORTID field shall contain the TransportID of the initiator port from which the command was received.

The INVALID MANAGEMENT IDENTIFIER KEY field shall be set to the value of the invalid management identifier key detected by the access controls coordinator.

NOTE 82 - The management identifier key is typically in the CDB for ACCESS CONTROL IN commands and in the parameter data for ACCESS CONTROL OUT commands.

#### 8.3.2.4.2.4 ACL LUN Conflicts access controls log portion page format

The ACL LUN Conflicts access controls log portion page (see table 527) contains details of logged ACL LUN conflicts (see 8.3.1.10) encountered by the access controls coordinator when a previously not-enrolled initiator port sends an ACCESS CONTROL OUT command with ACCESS ID ENROLL service action (see 8.3.3.4).

**Table 527 — ACL LUN Conflicts access controls log portion page format**

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)	TRANSPORTID ADDITIONAL LENGTH (m-32)						(LSB)
1								
2		Reserved						
3								
4	(MSB)	TIME STAMP						(LSB)
7								
8		TRANSPORTID						
m-1								
m	(MSB)	ACCESSID						(LSB)
m+23								

The TRANSPORTID ADDITIONAL LENGTH field indicates the additional length of the TRANSPORTID field beyond the minimum length of 24 bytes. The TransportID additional length shall be a multiple of four.

The TIME STAMP field shall contain zero or an indication of the time at which the ACCESS CONTROL OUT command with ACCESS ID ENROLL service action was processed as described in 8.3.1.10.

The TRANSPORTID field shall contain the TransportID of the initiator port from which the command was received that resulted in the ACL LUN conflict.

The ACCESSID field shall be set to the AccessID that the initiator port attempted to enroll. This shall correspond to an access identifier in ACL entry at the time the ACL LUN conflict event occurred.

### 8.3.2.5 REPORT OVERRIDE LOCKOUT TIMER service action

The ACCESS CONTROL IN command with REPORT OVERRIDE LOCKOUT TIMER service action (see table 528) is used query the value of the override lockout timer (see 8.3.1.8.2.2). If the ACCESS CONTROL IN command is implemented, the REPORT OVERRIDE LOCKOUT TIMER service action shall be implemented.

**Table 528 — ACCESS CONTROL IN command with REPORT OVERRIDE LOCKOUT TIMER service action**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (86h)							
1	Reserved			SERVICE ACTION (03h)				
2	(MSB)							
9	MANAGEMENT IDENTIFIER KEY							(LSB)
10	(MSB)							
13	ALLOCATION LENGTH							(LSB)
14	Reserved							
15	CONTROL							

If access controls are disabled, eight bytes of zeros shall be returned subject to the allocation length limitations described in 4.3.5.6 and GOOD status shall be returned.

If access controls are enabled and the contents of the MANAGEMENT IDENTIFIER KEY field do not match the current management identifier key (see 8.3.1.8) maintained by the access controls coordinator, then parameter data shall not be returned, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - INVALID MGMT ID KEY. This event shall be recorded in the invalid keys portion of the access controls log (see 8.3.1.10).

The ALLOCATION LENGTH field is defined in 4.3.5.6. The ALLOCATION LENGTH field value should be at least eight.

If access controls are enabled, the parameter data returned by the ACCESS CONTROL IN command with REPORT OVERRIDE LOCKOUT TIMER service action shall have the format shown in table 529.

**Table 529 — ACCESS CONTROL IN with REPORT OVERRIDE LOCKOUT TIMER parameter data**

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved							
1	Reserved							
2	(MSB)							
3	CURRENT OVERRIDE LOCKOUT TIMER							(LSB)
4	(MSB)							
5	INITIAL OVERRIDE LOCKOUT TIMER							(LSB)
6	(MSB)							
7	KEY OVERRIDES COUNTER							(LSB)

The CURRENT OVERRIDE LOCKOUT TIMER field shall be set to the current value of the override lockout timer (see 8.3.1.8.2.2).

The INITIAL OVERRIDE LOCKOUT TIMER field shall be set to the initial override lockout timer value (see 8.3.1.8.2.2) established by the last successful ACCESS CONTROL OUT command with MANAGE OVERRIDE LOCKOUT TIMER service action (see 8.3.3.7).

The KEY OVERRIDES COUNTER field shall be set to the value of the Key Override events counter in the access controls log (see 8.3.1.10).

### 8.3.2.6 REQUEST PROXY TOKEN service action

The ACCESS CONTROL IN command with REQUEST PROXY TOKEN service action (see table 530) is used to obtain a proxy token (see 8.3.1.6.2) for a logical unit to which that initiator port has non-proxy access rights. The returned proxy token may be used to pass temporary access to the logical unit to a third party that may use other proxy related service actions of the ACCESS CONTROL IN and ACCESS CONTROL OUT commands to gain access to the logical unit. If the ACCESS CONTROL IN command with REQUEST PROXY TOKEN service action is not supported, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

**Table 530 — ACCESS CONTROL IN command with REQUEST PROXY TOKEN service action**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (86h)							
1	Reserved			SERVICE ACTION (04h)				
2	(MSB) _____							
9	LUN VALUE							(LSB)
10	(MSB) _____							
13	ALLOCATION LENGTH							(LSB)
14	Reserved							
15	CONTROL							

If access controls are disabled, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

NOTE 83 - If access controls are disabled, all logical units are accessible and all initiator ports share the same LUN values for addressing. A proxy token is not needed because sharing LUN values is sufficient.

The LUN VALUE field shall contain the LUN value the application client uses to access the logical unit via the initiator port over which the proxy token is requested.

If the LUN value corresponds to a logical unit that is accessible to the requesting initiator port either through a TransportID or through the AccessID under which the initiator port is currently in the enrolled state (see 8.3.1.5.1), and the access controls coordinator has sufficient resources to create and manage a new proxy token, then the parameter data shown in table 531 shall be returned.

If the LUN value does not correspond to an accessible logical unit or corresponds to a logical unit accessible only through a proxy token, then the parameter data shall not be returned and the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - INVALID LU IDENTIFIER.

If the LUN value corresponds to a logical unit accessible only through an enrolled AccessID and the initiator port is in the pending-enrolled state, then the parameter data shall not be returned and the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - INITIATOR PENDING-ENROLLED.

If the access controls coordinator does not have enough resources to create and manage a new proxy token, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INSUFFICIENT ACCESS CONTROL RESOURCES.

The ALLOCATION LENGTH field is defined in 4.3.5.6. The ALLOCATION LENGTH field value should be at least eight.

The format of the parameter data returned by the ACCESS CONTROL IN command with REQUEST PROXY TOKEN service action is shown in table 531.

**Table 531 — ACCESS CONTROL IN with REQUEST PROXY TOKEN parameter data**

Bit Byte	7	6	5	4	3	2	1	0
0	PROXY TOKEN							
7								



### 8.3.3 ACCESS CONTROL OUT command

#### 8.3.3.1 ACCESS CONTROL OUT introduction

The service actions of the ACCESS CONTROL OUT command (see table 532) are used to request service actions by the access controls coordinator to limit or grant access to the logical units by initiator ports. If the ACCESS CONTROL OUT command is implemented, the ACCESS CONTROL IN command also shall be implemented. The ACCESS CONTROL OUT command shall not be affected by access controls.

**Table 532 — ACCESS CONTROL OUT service actions**

Service Action	Name	Type	Reference
00h	MANAGE ACL	M	8.3.3.2
01h	DISABLE ACCESS CONTROLS	M	8.3.3.3
02h	ACCESS ID ENROLL	M	8.3.3.4
03h	CANCEL ENROLLMENT	M	8.3.3.5
04h	CLEAR ACCESS CONTROLS LOG	M	8.3.3.6
05h	MANAGE OVERRIDE LOCKOUT TIMER	M	8.3.3.7
06h	OVERRIDE MGMT ID KEY	M	8.3.3.8
07h	REVOKE PROXY TOKEN	O	8.3.3.9
08h	REVOKE ALL PROXY TOKENS	O	8.3.3.10
09h	ASSIGN PROXY LUN	O	8.3.3.11
0Ah	RELEASE PROXY LUN	O	8.3.3.12
0Bh to 17h	Reserved		
18h to 1Fh	Vendor specific		
Key: M = Service action implementation is mandatory if ACCESS CONTROL OUT is implemented. O = Service action implementation is optional.			

The ACCESS CONTROL OUT command may be addressed to any logical unit whose standard INQUIRY data (see 6.4.2) has the ACC bit set to one (e.g., LUN 0), in which case it shall be processed in the same manner as if the command had been addressed to the ACCESS CONTROLS well known logical unit. If an ACCESS CONTROL OUT command is received by a device server whose standard INQUIRY data has the ACC bit set to zero, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID COMMAND OPERATION CODE.

If an ACCESS CONTROL OUT command is received while an IKEv2-SCSI CCS is in progress (see 5.14.4), the command shall be terminated with CHECK CONDITION status, with the sense key NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS. The sense key specific additional sense data may be set as described in 5.14.5.

The CDB format used by all ACCESS CONTROL OUT service actions is shown in table 533.

**Table 533 — ACCESS CONTROL OUT command format**

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (87h)							
1	Reserved			SERVICE ACTION (see table 532)				
2	Reserved							
9								
10	(MSB)	PARAMETER LIST LENGTH						(LSB)
13								
14	Reserved							
15	CONTROL							

The PARAMETER LIST LENGTH field indicates the amount of data being sent to the access controls coordinator in the Data-Out Buffer. The format of the parameter list is specific to each service action.

### 8.3.3.2 MANAGE ACL service action

#### 8.3.3.2.1 MANAGE ACL introduction

The ACCESS CONTROL OUT command with MANAGE ACL service action is used to authorize access or revoke access to a logical unit or logical units by initiator ports. The ACCESS CONTROL OUT command with MANAGE ACL service action adds, changes or removes an entry or multiple entries in the access controls coordinator's ACL (see 8.3.1.3). If the ACCESS CONTROL OUT command is implemented, the MANAGE ACL service action shall be implemented.

The format of the CDB for the ACCESS CONTROL OUT command with MANAGE ACL service action is shown in table 533 (see 8.3.3.1).

If the PARAMETER LIST LENGTH field in the CDB is zero, the access controls coordinator shall take no action and the command shall be completed with GOOD status.

If the value in the PARAMETER LIST LENGTH field is less than 20 or results in truncation of any ACE page (see table 535), then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

If the access controls coordinator is unable to complete the ACCESS CONTROL OUT command with MANAGE ACL service action because it has insufficient resources, then the access controls coordinator shall take no action and not change any of its state and the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INSUFFICIENT ACCESS CONTROL RESOURCES.

The format of the parameter data for the ACCESS CONTROL OUT command with MANAGE ACL service action is shown in table 534.

**Table 534 — ACCESS CONTROL OUT with MANAGE ACL parameter data format**

Bit Byte	7	6	5	4	3	2	1	0
	Parameter list header							
0	Reserved							
3								
4	(MSB)	MANAGEMENT IDENTIFIER KEY						(LSB)
11								
12	(MSB)	NEW MANAGEMENT IDENTIFIER KEY						(LSB)
19								
20	Reserved							
21	FLUSH	Reserved						
22	Reserved							
23	Reserved							
24	(MSB)	DLGENERATION						(LSB)
27								
	ACE pages							
28	ACE page 0							
	⋮							
n	ACE page x							

If access controls are enabled and the contents of the MANAGEMENT IDENTIFIER KEY field do not match the current management identifier key (see 8.3.1.8) maintained by the access controls coordinator, then the access controls coordinator's state shall not be altered, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - INVALID MGMT ID KEY. This event shall be recorded in the invalid keys portion of the access controls log (see 8.3.1.10).

If the contents of the MANAGEMENT IDENTIFIER KEY field match the current management identifier key maintained by the access controls coordinator, the access controls coordinator shall set its management identifier key to the value specified in the NEW MANAGEMENT IDENTIFIER KEY field and if access controls are disabled it shall enable them.

The FLUSH bit set to one instructs the access controls coordinator to place every initiator port in the enrolled state into the pending-enrolled state (see 8.3.1.5.1.4).

The DLGENERATION field specifies the DLgeneration value (see 8.3.1.4.4) associated with the default LUN values in the Grant/Revoke ACE pages in the parameter data.

The ACE pages that may follow the parameter list header provide additional changes to the ACL. Each ACE page describes one ACE in the ACL that is to be added, modified, or removed. The content and format of an ACE page is indicated by a page code (see table 535).

**Table 535 — ACE page codes**

Page Code	ACE Page Name	Reference
00h	Grant/Revoke	8.3.3.2.2
01h	Grant All	8.3.3.2.3
02h	Revoke Proxy Token	8.3.3.2.4
03h	Revoke All Proxy Tokens	8.3.3.2.5
04h to EFh	Reserved	
F0h to FFh	Vendor specific	

The following requirements apply to the processing of changes to the access control state:

- a) No change to the access control state shall occur if the ACCESS CONTROL OUT command with MANAGE ACL service action completes with a status other than GOOD status; and
- b) If the ACCESS CONTROL OUT command with MANAGE ACL service action completes with GOOD status, the following shall have been performed as a single indivisible event:
  - 1) Changes resulting from the contents of fields in the parameter list header shall be processed; and
  - 2) Changes resulting from the contents of ACE pages shall be processed;
    - a) Multiple ACE pages shall be processed sequentially;
    - b) If an ACE page contains conflicting instructions in LUACD descriptors, the instructions in the last LUACD descriptor within the ACE page shall take precedence; and
    - c) If an ACE containing an AccessID type access identifier (see 8.3.1.3.2) is replaced and the ACE page that caused the change has the NOCNCL bit (see 8.3.3.2.2) set to zero, then any initiator port in the enrolled state or pending-enrolled state under the AccessID in that ACE shall be placed in the not-enrolled state (see 8.3.1.5.1.2).

An ACE page contains conflicting instructions if either of the following is true:

- a) Two LUACD descriptors are present with the same LUN value and different default LUN values; or
- b) Two LUACD descriptors are present with different LUN values and the same default LUN value.

### 8.3.3.2.2 The Grant/Revoke ACE page

The Grant/Revoke ACE page (see table 536) is used to add, modify, or remove an ACE from the ACL (see 8.3.1.3).

**Table 536 — Grant/Revoke ACE page format**

Bit Byte	7	6	5	4	3	2	1	0
0	PAGE CODE (00h)							
1	Reserved							
2	(MSB)	PAGE LENGTH (n-3)						(LSB)
3								
4	NOCNCL	Reserved						
5	ACCESS IDENTIFIER TYPE							
6	(MSB)	ACCESS IDENTIFIER LENGTH (m-7)						(LSB)
7								
8	ACCESS IDENTIFIER							
m								
	LUACD descriptors							
m+1	LUACD descriptor 0							
m+20								
	⋮							
n-19	LUACD descriptor x							
n								

The PAGE LENGTH field specifies the number of additional bytes present in this ACE page.

A NOCNCL (no changes to current logical unit access) bit set to one specifies that the application client is telling the access controls coordinator that this ACE page makes no changes to the existing logical unit access conditions in the ACL. A NOCNCL bit set to zero specifies that the ACE page may or may not change existing logical unit access conditions. If the ACCESS IDENTIFIER TYPE specifies a TransportID (see 8.3.2.2.2.2), the NOCNCL bit shall be ignored.

The ACCESS IDENTIFIER TYPE and ACCESS IDENTIFIER LENGTH fields are described in 8.3.2.2.2.2.

The ACCESS IDENTIFIER field contains the identifier that the access controls coordinator uses to select the ACE that is to be added, modified, or removed. The format of the ACCESS IDENTIFIER field is specified in table 505 (see 8.3.1.13).

Any of the following conditions in the parameter header or any Grant/Revoke ACE page or Grant All ACE page shall cause the access coordinator to not change its state and shall cause the command to be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST:

- The value in the DLGENERATION field in the parameter list header (see 8.3.3.2.1) does not match the current value of the DLgeneration counter (see 8.3.1.4.4) maintained by the access controls coordinator;
- An ACCESS IDENTIFIER TYPE field specifies an unsupported value;

- c) An ACCESS IDENTIFIER TYPE field contains 01h (see 8.3.1.3.2) with an ACCESS IDENTIFIER field that contains an invalid TransportID (see 8.3.1.3.2) as defined for the applicable protocol standard;
- d) Two ACE pages that have the same values in the ACCESS IDENTIFIER TYPE and ACCESS IDENTIFIER fields; or
- e) Changes in the ACL that result in an ACL LUN conflict (see 8.3.1.5.2).

NOTE 84 - The application client is responsible for obtaining the current association of default LUN values to logical units and the DLgeneration value for that association prior to issuing this service action. The ACCESS CONTROL IN command with REPORT LU DESCRIPTORS service action (see 8.3.2.3) returns the necessary information.

Each LUACD descriptor (see table 537) describes the access to be allowed to one logical unit based on the access identifier in the ACE page. An ACE page may contain zero or more LUACD descriptors.

**Table 537 — ACE page LUACD descriptor format**

Bit Byte	7	6	5	4	3	2	1	0
0	ACCESS MODE							
1	Reserved							
3								
4	LUN VALUE							
11								
12	DEFAULT LUN							
19								

The ACCESS MODE field is described in 8.3.2.2.2.2.

The LUN VALUE field specifies the LUN value an accessing application client uses to access the logical unit via the initiator port to which the LUACD descriptor applies.

The DEFAULT LUN field specifies the logical unit to which the value in the LUN VALUE allows access. The DEFAULT LUN field shall contain a default LUN value (see 8.3.1.4.3). The value in the DEFAULT LUN field shall be consistent with the DLGENERATION field value specified in the parameter list header (see 8.3.3.2.1). If the DEFAULT LUN field references a well known logical unit, the access controls coordinator's state shall not be modified and the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the specified access mode is not supported or if the DEFAULT LUN field contains value that is not valid or the LUN VALUE field contains a value that the access controls coordinator does not support as a valid LUN, then the access controls coordinator's state shall not be modified and the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, the additional sense code set to ACCESS DENIED - INVALID LU IDENTIFIER, and the SENSE-KEY SPECIFIC field shall be set as described for the ILLEGAL REQUEST sense key in 4.5.2.4.2. If the error is an unsupported value in the LUN VALUE field, then the access controls coordinator should determine a suggested LUN value that is unlikely to produce an error while also minimizing the absolute value of the difference of the erroneous default LUN value and the suggested LUN value. If a suggested LUN value is determined, the first four bytes of the suggested LUN value shall be placed in the INFORMATION field and the last four bytes shall be placed in the COMMAND-SPECIFIC INFORMATION field of the sense data (see 4.5).

Based on the access identifier and the presence or absence of LUACD descriptors, the access controls coordinator shall add, modify, or remove an ACE in the ACL as shown in table 538.

**Table 538 — Access Coordinator Grant/Revoke ACE page actions**

		ACL already contains an ACE with the access identifier matching the one in the ACE page?	
		Yes	No
ACE page includes LUCAD descriptors?	Yes	Modify the existing ACE in the ACL.	Add a new ACE to the ACL.
	No	Remove the existing ACE from the ACL.	Take no action; this shall not be considered an error.

If the ACCESS IDENTIFIER TYPE indicates type AccessID, the enrollment state (see 8.3.1.5.1) of any initiator port that is enrolled under the specified AccessID, shall be affected as follows:

- a) If the ACE containing the AccessID is removed, the initiator port shall be placed in the not-enrolled state; or
- b) If the ACE containing the AccessID is modified by a Grant/Revoke ACE page or a Grant All ACE page, then;
  - A) If the NOCNCL bit is set to zero in that ACE page, the initiator port shall be placed in the not-enrolled state; or
  - B) If the NOCNCL bit is set to one in that ACE page, the enrollment state of the initiator port may be left unchanged or the initiator port may be placed in the not-enrolled state (see 8.3.1.5.1.2) based on vendor specific considerations.

### 8.3.3.2.3 The Grant All ACE page

The Grant All ACE page (see table 539) is used to add or modify an ACE in the ACL (see 8.3.1.3). An ACE added or modified using the Grant All ACE page allows initiator ports with the specified access identifier to access the SCSI target device as if access controls were disabled.

**Table 539 — Grant All ACE page format**

Bit Byte	7	6	5	4	3	2	1	0
0	PAGE CODE (01h)							
1	Reserved							
2	(MSB)PAGE LENGTH (n-3)(LSB)							
3								
4	NOCNCL	Reserved						
5	ACCESS IDENTIFIER TYPE							
6	(MSB)ACCESS IDENTIFIER LENGTH (m-7)(LSB)							
7								
8	ACCESS IDENTIFIER							
n								

The PAGE LENGTH, NOCNCL, ACCESS IDENTIFIER TYPE, ACCESS IDENTIFIER LENGTH, and ACCESS IDENTIFIER fields are defined in 8.3.3.2.2.

The Grant All ACE page shall be processed as if it is a Grant/Revoke ACE page (see 8.3.3.2.2) with one LUACD descriptor for every logical unit managed by the access controls coordinator with the fields in each LUACD containing:

- a) An access mode of 00h (see 8.3.2.2.2.2);
- b) A LUN VALUE field whose contents match the contents of the DEFAULT LUN field; and
- c) A DEFAULT LUN field whose contents reference the logical unit appropriate to the DLgeneration value (see 8.3.1.4.4).

8.3.3.2.4 The Revoke Proxy Token ACE page

The Revoke Proxy Token ACE page (see table 540) is used to revoke one or more proxy tokens (see 8.3.1.6.2).

Table 540 — Revoke Proxy Token ACE page format

Bit Byte	7	6	5	4	3	2	1	0
0	PAGE CODE (02h)							
1	Reserved							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
4	PROXY TOKEN 0							
11								
	⋮							
n-7	PROXY TOKEN x							
n								

The PAGE LENGTH field specifies the number of additional bytes present in this ACE page. If the page length is less than eight or not a multiple of eight, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense be set to PARAMETER LIST LENGTH ERROR.

The PROXY TOKEN field(s) specify the proxy tokens to be revoked. The access controls coordinator shall revoke each proxy token listed in a PROXY TOKEN field. If the contents of a PROXY TOKEN field do not identify a valid proxy token the field shall be ignored and this shall not be considered an error.

Multiple Revoke Proxy Token ACE pages may be included in the parameter data.



### 8.3.3.2.5 The Revoke All Proxy Tokens ACE page

The Revoke All Proxy Tokens ACE page (see table 541) is used to revoke all currently valid proxy tokens (see 8.3.1.6.2).

**Table 541 — Revoke All Proxy Tokens ACE page format**

Bit Byte	7	6	5	4	3	2	1	0
0	PAGE CODE (03h)							
1	Reserved							
2	(MSB)	PAGE LENGTH (0000h)						
3								(LSB)

Multiple Revoke All Proxy Tokens ACE pages may be included in the parameter data.

### 8.3.3.3 DISABLE ACCESS CONTROLS service action

The ACCESS CONTROL OUT command with DISABLE ACCESS CONTROLS service action is used to place the access controls coordinator in the access controls disabled state. If the ACCESS CONTROL OUT command is implemented, the DISABLE ACCESS CONTROLS service action shall be implemented.

The format of the CDB for the ACCESS CONTROL OUT command with DISABLE ACCESS CONTROLS service action is shown in table 533 (see 8.3.3.1).

If access controls are disabled or if the PARAMETER LIST LENGTH field in the CDB is zero, the access controls coordinator shall take no action and the command shall be completed with GOOD status.

If the value in the PARAMETER LIST LENGTH field is neither zero nor 12, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

If the value in the PARAMETER LIST LENGTH field is 12, the parameter list shall have the format shown in table 542.

**Table 542 — ACCESS CONTROL OUT with DISABLE ACCESS CONTROLS parameter data format**

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved							
3								
4	(MSB)	MANAGEMENT IDENTIFIER KEY						
11								(LSB)

If access controls are enabled and the contents of the MANAGEMENT IDENTIFIER KEY field do not match the current management identifier key (see 8.3.1.8) maintained by the access controls coordinator, then the access controls coordinator's states shall not be altered, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - INVALID MGMT ID KEY. This event shall be recorded in the invalid keys portion of the access controls log (see 8.3.1.10).

In response to an ACCESS CONTROL OUT command with DISABLE ACCESS CONTROLS service action with correct management identifier key value the access controls coordinator shall:

- a) Disable access controls;
- b) Clear the ACL (see 8.3.1.3);
- c) Place all initiator ports into the not-enrolled state (see 8.3.1.5.1);
- d) Set the management identifier key to zero (see 8.3.1.8);
- e) Set the override lockout timer to zero (see 8.3.1.8.2.2);
- f) Set the initial override lockout timer value to zero (see 8.3.1.8.2.2);
- g) Clear the access controls log, including resetting the events counters to zero, with the exception of the key overrides portion of the access controls log (see 8.3.1.10);
- h) Allow all initiator port's access to all logical units at their default LUN value;
- i) Optionally, set the DLgeneration counter to zero (see 8.3.1.4.4); and
- j) Establish a unit attention condition for the initiator port associated with every I\_T nexus in each logical unit in the SCSI target device, with the additional sense code set to REPORTED LUNS DATA HAS CHANGED.

**8.3.3.4 ACCESS ID ENROLL service action**

The ACCESS ID ENROLL service action of the ACCESS CONTROL OUT command is used by an application client to enroll an AccessID for an initiator port with the access controls coordinator. If the ACCESS CONTROL OUT command is implemented, the ACCESS ID ENROLL service action shall be implemented.

The format of the CDB for the ACCESS CONTROL OUT command with ACCESS ID ENROLL service action is shown in table 533 (see 8.3.3.1).

If access controls are disabled or if the PARAMETER LIST LENGTH field in the CDB is zero, the access controls coordinator shall take no action and the command shall be completed with GOOD status.

If the value in the PARAMETER LIST LENGTH field is neither zero nor 24, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

If the value in the PARAMETER LIST LENGTH field is 24, the parameter list shall have the format shown in table 543.

**Table 543 — ACCESS CONTROL OUT with ACCESS ID ENROLL parameter data format**

Bit Byte	7	6	5	4	3	2	1	0
0	ACCESSID							
15								
16	Reserved							
23								

The ACCESSID field is described in 8.3.1.3.2.

If the initiator port is in the enrolled state or pending-enrolled state (see 8.3.1.5.1) under a specific AccessID and the ACCESSID field contains a different AccessID, then the access controls coordinator shall place the initiator port in the pending-enrolled state, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - ENROLLMENT CONFLICT.

If the initiator port is in the enrolled state or pending-enrolled state under a specific AccessID and the ACCESSID field contains a matching AccessID, the access controls coordinator shall place the initiator port in the enrolled state and make no other changes.

If the initiator port is in the not-enrolled state and the ACCESSID field contents do not match the AccessID in any ACE in the ACL (see 8.3.1.3), then the initiator port shall remain in the not-enrolled state and the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - NO ACCESS RIGHTS.

If the initiator port is in the not-enrolled state and the ACCESSID field contents matches the AccessID in an ACE in the ACL, the actions taken depend on whether enrolling the initiator port would create an ACL LUN conflict (see 8.3.1.5.2). If there is no ACL LUN conflict, the initiator port shall be placed in the enrolled state (see 8.3.1.5.1.3). If there is an ACL LUN conflict, then the initiator port shall remain in the not-enrolled state and the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - ACL LUN CONFLICT. This event shall be recorded in the ACL LUN conflicts portion of the access controls log (see 8.3.1.10).

An application client that receives the ACCESS DENIED - ACL LUN CONFLICT additional sense code should remove any proxy access rights it has acquired using the ACCESS CONTROL OUT command with RELEASE PROXY LUN service action and retry the enrollment request. If the ACL LUN conflict resulted from proxy access, the retried enrollment succeeds. Otherwise, the mechanisms for resolving ACL LUN conflicts are outside the scope of this standard.

#### **8.3.3.5 CANCEL ENROLLMENT service action**

The ACCESS CONTROL OUT command with CANCEL ENROLLMENT service action is used to remove an initiator port's enrollment with the access controls coordinator (see 8.3.1.5). Successful completion of this command changes the state of the initiator port to the not-enrolled state. If the ACCESS CONTROL OUT command is implemented, the CANCEL ENROLLMENT service action shall be implemented.

The ACCESS CONTROL OUT command with CANCEL ENROLLMENT service action should be used by an application client prior to any period where use of its accessible logical units may be suspended for a lengthy period of time (e.g., when a host is preparing to shutdown). This allows the access controls coordinator to free any resources allocated to manage the enrollment for the initiator port.

The format of the CDB for the ACCESS CONTROL OUT command with CANCEL ENROLLMENT service action is shown in table 533 (see 8.3.3.1).

If access controls are disabled, the access controls coordinator shall take no action and the command shall be completed with GOOD status.

There is no parameter data for the ACCESS CONTROL OUT command with CANCEL ENROLLMENT service action. If the PARAMETER LIST LENGTH field in the CDB is not set to zero, the initiator port's enrollment shall not be changed and the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

If the PARAMETER LIST LENGTH field in the CDB is set to zero, the initiator port shall be placed in the not-enrolled state (see 8.3.1.5.1.2). Any subsequent commands addressed to the logical units no longer accessible are handled according to the requirements stated in 8.3.1.7.

### 8.3.3.6 CLEAR ACCESS CONTROLS LOG service action

The ACCESS CONTROL OUT command with CLEAR ACCESS CONTROLS LOG service action is used to instruct the access controls coordinator to reset a specific access control events counter to zero and to clear a portion of the access controls log (see 8.3.1.10). If the ACCESS CONTROL OUT command is implemented, the CLEAR ACCESS CONTROLS LOG service action shall be implemented.

The format of the CDB for the ACCESS CONTROL OUT command with CLEAR ACCESS CONTROLS LOG service action is shown in table 533 (see 8.3.3.1).

If access controls are disabled or if the PARAMETER LIST LENGTH field in the CDB is zero, the access controls coordinator shall take no action and the command shall be completed with GOOD status.

If the value in the PARAMETER LIST LENGTH field is neither zero nor 12, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

If the value in the PARAMETER LIST LENGTH field is 12, the parameter list shall have the format shown in table 544.

**Table 544 — ACCESS CONTROL OUT with CLEAR ACCESS CONTROLS LOG parameter data format**

Bit Byte	7	6	5	4	3	2	1	0		
0	Reserved									
2										
3	Reserved						LOG PORTION			
4	(MSB)									
11	MANAGEMENT IDENTIFIER KEY									
	(LSB)									

The LOG PORTION field (see table 545) specifies the access controls log portion to be cleared.

**Table 545 — CLEAR ACCESS CONTROLS LOG LOG PORTION field**

Code	Description
00b	Reserved
01b	Invalid Keys portion
10b	ACL LUN Conflicts portion
11b	Reserved

If access controls are enabled and the contents of the MANAGEMENT IDENTIFIER KEY field do not match the current management identifier key (see 8.3.1.8) maintained by the access controls coordinator, then the access controls coordinator's states shall not be altered, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - INVALID MGMT ID KEY. This event shall be recorded in the invalid keys portion of the access controls log (see 8.3.1.10).

In response to an ACCESS CONTROL OUT command with CLEAR ACCESS CONTROLS LOG service action with correct management identifier key value the access controls coordinator shall perform the following to clear the portion of the access controls log identified by the LOG PORTION field (see table 545) in the parameter data:

- Set the events counter for the specified log portion to zero; and
- If the specified log portion contains log records, remove the log records from the specified log portion.

### 8.3.3.7 MANAGE OVERRIDE LOCKOUT TIMER service action

The ACCESS CONTROL OUT command with MANAGE OVERRIDE LOCKOUT TIMER service action is used to manage the override lockout timer (see 8.3.1.8.2.2). If the ACCESS CONTROL OUT command is implemented, the MANAGE OVERRIDE LOCKOUT TIMER service action shall be implemented.

If access controls are disabled, the access controls coordinator shall take no action and the command shall be completed with GOOD status.

The format of the CDB for the ACCESS CONTROL OUT command with MANAGE OVERRIDE LOCKOUT TIMER service action is shown in table 533 (see 8.3.3.1).

If the PARAMETER LIST LENGTH field in the CDB is zero, the access controls coordinator shall reset the override lockout timer to the current initial override lockout timer value maintained by the access controls coordinator.

If the value in the PARAMETER LIST LENGTH field is neither zero nor 12, the device server shall respond with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

If the value in the PARAMETER LIST LENGTH field is 12, the parameter list shall have the format shown in table 546.

**Table 546 — ACCESS CONTROL OUT with MANAGE OVERRIDE LOCKOUT TIMER parameter data format**

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved							
1								
2	(MSB)	NEW INITIAL OVERRIDE LOCKOUT TIMER						
3								(LSB)
4	(MSB)	MANAGEMENT IDENTIFIER KEY						
11								(LSB)

The NEW INITIAL OVERRIDE LOCKOUT TIMER field specifies the value that access controls coordinator shall maintain for initial override lockout timer if the specified management identifier key is correct.

If access controls are enabled and the contents of the MANAGEMENT IDENTIFIER KEY field do not match the current management identifier key (see 8.3.1.8) maintained by the access controls coordinator, then the access controls coordinator shall not change the initial override lockout timer value but shall set the override lockout timer to the unaltered current initial override lockout timer value. The command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - INVALID MGMT ID KEY. This event shall be recorded in the invalid keys portion of the access controls log (see 8.3.1.10).

In response to an ACCESS CONTROL OUT command with MANAGE OVERRIDE LOCKOUT TIMER service action with correct management identifier key value the access controls coordinator shall:

- Replace the currently saved initial override lockout timer with the value in the NEW INITIAL OVERRIDE LOCKOUT TIMER field; and
- Set the override lockout timer to the new initial value.

8.3.3.8 OVERRIDE MGMT ID KEY service action

The ACCESS CONTROL OUT command with OVERRIDE MGMT ID KEY service action is used to override the current management identifier key (see 8.3.1.4.2) maintained by the access controls coordinator. The ACCESS CONTROL OUT command with OVERRIDE MGMT ID KEY service action should be used in a failure situation where the application client no longer has access to its copy of the current management identifier key.

Successful use of the ACCESS CONTROL OUT command with OVERRIDE MGMT ID KEY service action is restricted by the override lockout timer (see 8.3.1.8.2.2).

If the ACCESS CONTROL OUT command is implemented, the OVERRIDE MGMT ID KEY service action shall be implemented.

The format of the CDB for the ACCESS CONTROL OUT command with OVERRIDE MGMT ID KEY service action is shown in table 533 (see 8.3.3.1).

If access controls are disabled or if the PARAMETER LIST LENGTH field in the CDB is zero, the access controls coordinator shall take no action and the command shall be completed with GOOD status.

If access controls are enabled, the access controls coordinator shall log every ACCESS CONTROL OUT command with OVERRIDE MGMT ID KEY service action processed whether successful or not in the access controls log as specified in 8.3.1.10.

If the value in the PARAMETER LIST LENGTH field is neither zero nor 12, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

If the value in the PARAMETER LIST LENGTH field is 12, the parameter data shall have the format shown in table 547.

Table 547 — ACCESS CONTROL OUT with OVERRIDE MGMT ID KEY parameter data format

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved							
3								
4	(MSB)	NEW MANAGEMENT IDENTIFIER KEY						
11								(LSB)

The NEW MANAGEMENT IDENTIFIER KEY field specifies a new management identifier key.

If the override lockout timer managed by the access controls coordinator is not zero, the access controls coordinator's states shall not be altered, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

If the override lockout timer managed by the access controls coordinator is zero, then the access controls coordinator shall replace the current management identifier key with the value in the to the NEW MANAGEMENT IDENTIFIER KEY field.

### 8.3.3.9 REVOKE PROXY TOKEN service action

An application client for an initiator port uses the ACCESS CONTROL OUT command with REVOKE PROXY TOKEN service action to cancel all proxy access rights to a logical unit that have been granted under the specified proxy token (see 8.3.1.6.2). If this service action is not supported, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The format of the CDB for the ACCESS CONTROL OUT command with REVOKE PROXY TOKEN service action is shown in table 533 (see 8.3.3.1).

If access controls are disabled or if the PARAMETER LIST LENGTH field in the CDB is zero, the access controls coordinator shall take no action and the command shall be completed with GOOD status.

If the value in the PARAMETER LIST LENGTH field is neither zero nor eight, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

If the value in the PARAMETER LIST LENGTH field is eight, the parameter data shall have the format shown in table 548.

**Table 548 — ACCESS CONTROL OUT with REVOKE PROXY TOKEN parameter data format**

Bit Byte	7	6	5	4	3	2	1	0
0	PROXY TOKEN							
7								

If the PROXY TOKEN field does not contain a valid proxy token previously obtained via the initiator port, no action is taken by the access controls coordinator. This shall not be considered an error.

If the proxy token is valid, the access controls coordinator shall take the following actions:

- a) Invalidate the proxy token; and
- b) Deny access to the associated logical unit by any initiator port whose rights were granted under that proxy token via an ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action (see 8.3.3.11) according to the requirements stated in 8.3.1.7.

### 8.3.3.10 REVOKE ALL PROXY TOKENS service action

An application client for an initiator port uses the ACCESS CONTROL OUT command with REVOKE ALL PROXY TOKENS service action to cancel all proxy access rights to a specified logical unit that it obtained with zero or more proxy tokens (see 8.3.1.6.2). If this service action is not supported, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The format of the CDB for the ACCESS CONTROL OUT command with REVOKE ALL PROXY TOKENS service action is shown in table 533 (see 8.3.3.1).

If access controls are disabled or if the PARAMETER LIST LENGTH field in the CDB is zero, the access controls coordinator shall take no action and the command shall be completed with GOOD status.

If the value in the PARAMETER LIST LENGTH field is neither zero nor eight, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

If the value in the PARAMETER LIST LENGTH field is eight, the parameter data shall have the format shown in table 549.

**Table 549 — ACCESS CONTROL OUT with REVOKE ALL PROXY TOKENS parameter data format**

Bit Byte	7	6	5	4	3	2	1	0
0	LUN VALUE							
7								

If the LUN in the LUN VALUE field is not associated to a logical unit to which the requesting initiator port has non-proxy access rights based on the contents of an ACE (see 8.3.1.3) or if the LUN value is based on a proxy token (see 8.3.1.6.2), then no further action is taken by the access controls coordinator. This shall not be considered an error.

If the LUN value is associated to a logical unit to which the requesting initiator port has non-proxy access rights, the access controls coordinator shall take the following additional actions:

- a) Invalidate all proxy tokens for the initiator port for the logical unit specified by the LUN VALUE field;
- b) Deny access to that logical unit by any initiator port whose rights were granted under any of the invalidated proxy tokens via an ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action (see 8.3.3.11) according to the requirements stated in 8.3.1.7.

**8.3.3.11 ASSIGN PROXY LUN service action**

The ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action is used to request access to a logical unit under the rights of a proxy token (see 8.3.1.6.2) and to assign that logical unit a particular LUN value for addressing by the requesting initiator port. If this service action is not supported, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The format of the CDB for the ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action is shown in table 533 (see 8.3.3.1).

If the PARAMETER LIST LENGTH field in the CDB is zero, the access controls coordinator shall take no action and the command shall be completed with GOOD status.

If the value in the PARAMETER LIST LENGTH field is neither zero nor 16, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.



If the value in the PARAMETER LIST LENGTH field is 16, the parameter data shall have the format shown in table 550.

**Table 550 — ACCESS CONTROL OUT with ASSIGN PROXY LUN parameter data format**

Bit Byte	7	6	5	4	3	2	1	0
0	PROXY TOKEN							
7								
8	LUN VALUE							
15								

The PROXY TOKEN field contains a proxy token. If the contents of the PROXY TOKEN field are not valid, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - INVALID PROXY TOKEN.

NOTE 85 - If access controls are disabled, there are no valid proxy tokens and the device server always responds with the specified error information. This differs from the behavior of many other ACCESS CONTROL OUT service actions where the response is GOOD status when access controls are disabled. The difference in behavior is intended to inform the application client that its request for the new LUN assignment failed.

The LUN VALUE field specifies the LUN value the application client intends to use when accessing the logical unit described by the proxy token.

If the proxy token is valid but the access controls coordinator is unable to assign the requested LUN value to the associated logical unit (e.g., because the LUN value already is associated with a logical unit for the initiator port, or because the LUN value is not a supported logical unit address), then access rights shall not be granted, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - INVALID LU IDENTIFIER, and the SENSE-KEY SPECIFIC field shall be set as described for the ILLEGAL REQUEST sense key in 4.5.2.4.2. The access controls coordinator should determine a suggested LUN value that is unlikely to produce an error while also minimizing the absolute value of the difference of the erroneous default LUN value and the suggested LUN value. If a suggested LUN value is determined, the first four bytes of the suggested LUN value shall be placed in the INFORMATION field and the last four bytes shall be placed in the COMMAND-SPECIFIC INFORMATION field of the sense data (see 4.5).

If the proxy token is valid but the access controls coordinator has insufficient resources to manage proxy logical unit access, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INSUFFICIENT ACCESS CONTROL RESOURCES.

If the proxy token is valid and the access controls coordinator has sufficient resources, the initiator port shall be allowed proxy access to the referenced logical unit at the specified LUN value.

### 8.3.3.12 RELEASE PROXY LUN service action

The ACCESS CONTROL OUT command with RELEASE PROXY LUN service action is used to release proxy access to a logical unit created with a proxy token (see 8.3.1.6.2) and the ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action (see 8.3.3.11). If this service action is not supported, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The ACCESS CONTROL OUT command with RELEASE PROXY LUN service action should be used when an application client no longer requires the logical unit access rights granted to an initiator port under a proxy token (e.g., when a copy manager has completed a specific third party copy operation under a proxy token). This allows the access controls coordinator to free any resources allocated to manage the proxy access.

The format of the CDB for the ACCESS CONTROL OUT command with RELEASE PROXY LUN service action is shown in table 533 (see 8.3.3.1).

If the PARAMETER LIST LENGTH field in the CDB is zero, the access controls coordinator shall take no action and the command shall be completed with GOOD status.

If the value in the PARAMETER LIST LENGTH field is neither zero nor eight, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

If the value in the PARAMETER LIST LENGTH field is eight, the parameter data shall have the format shown in table 551.

Table 551 — ACCESS CONTROL OUT with RELEASE PROXY LUN parameter data format

Bit Byte	7	6	5	4	3	2	1	0
0	LUN VALUE							
7								

The LUN VALUE field specifies a LUN value that was associated with a logical unit based on a proxy token using an ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action. If the LUN value was not assigned to a logical unit by an ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

NOTE 86 - If access controls are disabled, there are no valid proxy tokens and therefore no LUN value could be assigned to a logical unit by an ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action so the device server always responds with the specified error information. This differs from the behavior of many other ACCESS CONTROL OUT service actions where the response is GOOD status when access controls are disabled. The difference in behavior is intended to inform the application client that the LUN value remains as a valid address for the logical unit.

If the LUN value was assigned to a logical unit by an ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action, the access controls coordinator shall not allow access to the logical unit at the specified LUN value.

## 8.4 TARGET LOG PAGES well known logical unit

The TARGET LOG PAGES well known logical unit shall only process the commands listed in table 552. If a command is received by the TARGET LOG PAGES well known logical unit that is not listed in table 552, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID COMMAND OPERATION CODE.

**Table 552 — Commands for the TARGET LOG PAGES well known logical unit**

Command name	Operation code	Type	Reference
INQUIRY	12h	M	6.4
LOG SELECT	4Ch	M	6.5
LOG SENSE	4Dh	M	6.6
REQUEST SENSE	03h	M	6.29
TEST UNIT READY	00h	M	6.33
Key: M = Command implementation is mandatory.			

The TARGET LOG PAGES well known logical unit shall support the Protocol Specific Port log page (see 7.2.10) and may support other log pages with parameters that apply to the SCSI target device.

## 8.5 SECURITY PROTOCOL well known logical unit

The SECURITY PROTOCOL well known logical unit shall only process the commands listed in table 553. If a command is received by the SECURITY PROTOCOL well known logical unit that is not listed in table 553, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID COMMAND OPERATION CODE.

**Table 553 — Commands for the SECURITY PROTOCOL well known logical unit**

Command name	Operation code	Type	Reference
INQUIRY	12h	M	6.4
REQUEST SENSE	03h	M	6.29
SECURITY PROTOCOL IN	A2h	M	6.30
SECURITY PROTOCOL OUT	B5h	M	6.31
TEST UNIT READY	00h	M	6.37
Key: M = Command implementation is mandatory.			

## 8.6 MANAGEMENT PROTOCOL well known logical unit

The MANAGEMENT PROTOCOL well known logical unit shall only process the commands listed in table 554. If a command is received by the MANAGEMENT PROTOCOL well known logical unit that is not listed in table 554, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID COMMAND OPERATION CODE.

**Table 554 — Commands for the MANAGEMENT PROTOCOL well known logical unit**

Command name	Operation code	Type	Reference
INQUIRY	12h	M	6.4
MANAGEMENT PROTOCOL IN	A3h/10h <sup>a</sup>	M	6.7
MANAGEMENT PROTOCOL OUT	A4h/10h <sup>a</sup>	M	6.8
REQUEST SENSE	03h	M	6.29
TEST UNIT READY	00h	M	6.37
Key: M = Command implementation is mandatory.			
<sup>a</sup> This command is defined by a combination of operation code and service action. The operation code value is shown preceding the slash and the service action value is shown after the slash.			

## 9 Security manager command set

A security manager shall only process the commands listed in table 555. If a command is received by the security manager that is not listed in table 555, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID COMMAND OPERATION CODE.

**Table 555 — Commands for a security manager**

Command name	Operation code	Type	Reference
INQUIRY	12h	M	6.4
RECEIVE CREDENTIAL	7Fh/1800 <sup>a</sup>	M	6.19
REPORT LUNS	A0h	M	6.23
REQUEST SENSE	03h	M	6.29
SECURITY PROTOCOL IN	A2h	M	6.30
SECURITY PROTOCOL OUT	B5h	M	6.31
TEST UNIT READY	00h	M	6.37
Type Key: M = Command implementation is mandatory.			
<sup>a</sup> This command is defined by a combination of operation code and service action. The operation code value is shown preceding the slash and the service action value is shown after the slash.			

## **Annex A**

(informative)

### **Terminology mapping**

The introduction of a model for SCSI devices with multiple ports resulted in changes in terminology between SPC-2 and this standard (see table A.1).

**Table A.1 — This standard to SPC-2 terminology mapping**

<b>SPC-3 equivalent term</b>	<b>SPC-2 term</b>
initiator port identifier	initiator identifier
queue	task set
SCSI initiator port	initiator
SCSI port	port
SCSI port identifier	device identifier
SCSI port identifier	SCSI identifier
SCSI target port	target
target port identifier	target identifier

## Annex B

(Informative)

### PERSISTENT RESERVE IN/OUT functionality for RESERVE/RELEASE replacement

#### B.1 Introduction

This annex specifies the PERSISTENT RESERVE OUT command features necessary to replace the reserve/release management method (see SPC-2) and provides guidance on how to perform a third party reservation using persistent reservations. The PERSISTENT RESERVE IN command is not used to replace any feature of the reserve/release management method.

#### B.2 Replacing the reserve/release method with the PERSISTENT RESERVE OUT COMMAND

The minimum PERSISTENT RESERVE OUT command (see 6.14) features necessary to replace the reserve/release management method (see SPC-2) are shown in table B.1.

**Table B.1 — PERSISTENT RESERVE OUT command features**

PERSISTENT RESERVE OUT command features		Replaces reserve/release
Service Action	REGISTER	Yes <sup>b</sup>
	RESERVE	Yes
	RELEASE	Yes
	CLEAR	Yes <sup>c</sup>
	PREEMPT	No
	PREEMPT AND ABORT	No
	REGISTER AND IGNORE EXISTING KEY	Yes <sup>b</sup>
	REGISTER AND MOVE	Yes <sup>d</sup>
Scope	LU_SCOPE	Yes
Type	Write Exclusive	No
	Exclusive Access	Yes
	Write Exclusive – Registrants Only	No
	Exclusive Access – Registrants Only	No
	Write Exclusive – All Registrants	No
	Exclusive Access – All Registrants	No
<sup>b</sup> An implementation uses either the REGISTER or REGISTER AND IGNORE EXISTING KEY service action. <sup>c</sup> Necessary to clear the registration and reservation (e.g, a failed initiator). <sup>d</sup> Necessary only for third party reservations.		

### B.3 Third party reservations

For some uses of the EXTENDED COPY command (see 6.3), the application client performs a locking function to maintain data integrity on the source and may also lock the destination device prior to starting the copy operation. The persistent reservation management method may be used to perform the locking function. Other methods (e.g., access controls, see 8.3) may also perform the locking function.

To accomplish a third party persistent reservation the following steps are recommended:

- 1) Backup application uses the REGISTER service action to register an I\_T nexus with a logical unit (e.g., a tape drive logical unit);
- 2) Backup application uses the RESERVE service action to establish a persistent reservation with the Exclusive Access type;
- 3) Backup application prepares the logical unit for access (e.g., medium is loaded and positioned);
- 4) Backup application uses the REGISTER AND MOVE service action to register the I\_T nexus that the copy manager is expected to use and to move the persistent reservation to that I\_T nexus;
- 5) Backup application sends the EXTENDED COPY command to the copy manager that includes a third party persistent reservations source I\_T nexus segment descriptor (see 6.3.7.19);
- 6) Copy manager processes all segment descriptors in the received EXTENDED COPY command except the third party persistent reservations source I\_T nexus segment descriptor; and
- 7) Copy manager issues a REGISTER AND MOVE service action, using the reservation key and I\_T nexus specified in the third party persistent reservations source I\_T nexus segment descriptor received from the backup application (see step 5), to move the persistent reservation back to the original I\_T nexus.



## Annex C

(Informative)

### Variations between this standard and equivalent security protocols

#### C.1 IKEv2 protocol details and variations for IKEv2-SCSI

The IKEv2 protocol details and variations specified in RFC 4306 apply to IKEv2-SCSI (i.e., this standard) as follows:

- a) Any SECURITY PROTOCOL OUT command with a transfer length of up to 16 384 bytes is not terminated with an error due to the number of bytes transferred;
- b) The timeout and retransmission mechanisms defined in RFC 4306 are not used by this standard, instead a new payload is defined to transfer appropriate timeout values to the device server;
- c) Each SCSI command used by this standard completes by conveying a status from the device server to the application client;
- d) The IKEv2 header EXCHANGE TYPE field is reserved in this standard as a result of equivalent information being transferred in the SECURITY PROTOCOL OUT command and SECURITY PROTOCOL IN command CDBs;
- e) The IKEv2 header VERSION bit is reserved in this standard;
- f) This standard uses the pseudo-random functions (PRF) functions defined by RFC 4306;
- g) The key derivation functions defined and used by this standard (see 5.14.3) are equivalent to the PRF+ found in RFC 4306;
- h) The SA creation transactions (see 3.1.123) defined by this standard are not overlapped. If an application client attempts to start a second SA creation transaction before the first is completed, the offending command is terminated as described in 5.14.4.1, but this does not affect the SA creation transaction that is already in progress;
- i) The NO\_PROPOSAL\_CHOSEN and INVALID\_KEY\_PAYLOAD notify error types are replaced by the SA CREATION PARAMETER VALUE INVALID additional sense code (see 7.6.3.9) because IKEv2-SCSI has a different negotiation structure. As defined in RFC 4306, an IKEv2 initiator offers one or more proposals to a responder without knowing what is acceptable to the responder, and chooses a DH group without knowing whether it is acceptable to the responder. These two notify error types allow the responder to inform the initiator that one or more of its choices are not acceptable. In contrast, an IKEv2-SCSI application client obtains the device server capabilities in the Device Capabilities step (see 5.14.4.7) and selects algorithms from them in the Key Exchange step (see 5.14.4.8). An error only occurs if the application client has made an invalid selection, hence the SA CREATION PARAMETER VALUE INVALID description;
- j) IKEv2 version numbers (see RFC 4306) are used by this standard (see 7.6.3.4), but the ability to respond to an unsupported version number with the highest version number to be used is not supported, and this standard does not include checks for version downgrade attacks;
- k) IKEv2 cookies (see RFC 4306) are not used by this standard;
- l) IKEv2 cryptographic algorithm negotiation (see RFC 4306) is replaced by the Device Server Capabilities step (see 5.14.4.7) and the Key Exchange step (see 5.14.4.8) (i.e., the IKEv2 proposal construct is not used by this standard);
- m) In this standard an SA is rekeyed by replacing it with a new SA:
  - A) CHILD\_SAs are not used by this standard;
  - B) The RFC 4306 discussion of CHILD\_SAs does not apply to this standard;
  - C) Coexistence of the original SA and the new SA is achieved for rekeying purposes by restricting the device server's ability to delete SAs to the following cases:
    - a) Expiration of a timeout (see 7.6.3.5.14);
    - b) Processing of an IKEv2-SCSI Delete function (see 5.14.4.13); and
    - c) Responding to an initial contact notification (see 7.6.3.5.8);
 and

- D) IKEv2 does not support rekeying notification for IKE\_SAs, therefore this standard does not support rekeying notification;
- n) The choice of authentication methods for both transfer directions is negotiated using the SA\_AUTH\_OUT IKEv2-SCSI cryptographic algorithm descriptor and SA\_AUTH\_IN IKEv2-SCSI cryptographic algorithm descriptor (see 7.6.3.6.6) during the Key Exchange step (see 5.14.4.8);
- o) The usage of Certificate Encodings in the Certificate payload and Certificate Request payload (see 7.6.3.5.5) are constrained as follows:
  - A) In accordance with the recommendations in RFC 4718, Certificate Encoding values 01h-03h and 05h-0Ah are prohibited;
  - B) This standard forbids the use of URL-based Certificate Encodings (i.e., Certificate Encodings values 0Ch and 0Dh); and
  - C) Certificate Encoding values that RFC 4306 defines as vendor specific are reserved in this standard;
- p) Deleting an SA requires knowing the SAs (i.e., SPIs) in both directions and including both SAs in the Delete payload. The RFC 4306 description of the Delete payload is vague enough to allow this. The requirement is consistent with the SCSI model for SAs;
- q) The Vendor ID payload is not used by this standard;
- r) Traffic Selectors (see RFC 4306) are not used by this standard;
- s) The requirements in RFC 4306 on nonces are to be followed for the random nonces (see 3.1.115) defined by this standard;
- t) The RFC 4306 requirements on address and port agility are specific to the user datagram protocol and the IP protocol and do not apply to this standard;
- u) Keys for the Authentication step are generated as specified in RFC 4306;
- v) This standard uses a slightly modified version of the authentication calculations in RFC 4306 (see 7.6.3.5.6);
- w) The RFC 4306 sections that describe the following features are not used by this standard:
  - A) Extensible authentication protocol methods;
  - B) Generating keying Material for CHILD\_SAs;
  - C) Rekeying an IKE SA using CREATE\_CHILD\_SA;
  - D) Requesting an internal address;
  - E) Requesting the peer's version;
  - F) IPComp;
  - G) NAT traversal; and
  - H) Explicit congestion notification;
- x) IKEv2 Error Handling (see RFC 4306) is replaced by the use of CHECK CONDITION status and sense data by this standard. See 7.6.3.9 for details of how errors reported in the Notify payload are translated to sense data;
- y) The description of how combined mode algorithms are used in the Encrypted payload in this standard predates the definition of equivalent functionality in IETF standards. IETF standards omit the Integrity transform instead of using AUTH\_COMBINED;
- z) The command-response architecture of SCSI makes it difficult to protect the device server against denial of service attacks, and no such protection is defined by this standard. Protection against denial of service attacks against the application client is described in 7.6.3.8; and
- aa) IKEv2-SCSI requires Encrypted Payloads be padded to at least the 4 byte minimum alignment required by ESP-SCSI, whereas IKEv2 imposes no such requirement.

Where this standard uses IKE payload names (see 7.6.3.4) RFC 4306 uses the shorthand notation shown in table C.1.

**Table C.1 — IKE payload names shorthand**

<b>IKE payload name in this standard <sup>a</sup></b>	<b>RFC 4306 shorthand <sup>b</sup></b>
Security Association	SAi or SAr
Key Exchange	KEi or KEr
Identification – Application Client	IDi
Identification – Device Server	IDr
Certificate	CERTi or CERTr
Certificate Request	CERTREQi
Authentication	AUTHi or AUTHr
Nonce	NONCEi or NONCEr
Notify	N-ICi or N-ICr
Delete	Di
Vendor ID	Vi or Vr
Traffic Selector – Application Client	TSi
Traffic Selector – Device Server	TSr
Encrypted	Ei or Er
Configuration	CPi or CPr
Extensible Authentication	EAPi or EAPr
<sup>a</sup> To facilitate future enhancements, all IKE payloads are listed in this table, but not all entries in this table are used in this standard. <sup>b</sup> In RFC 4306 the lowercase i indicates initiator and r indicates responder. In this standard, the initiator is the application client and all such IKE payloads (e.g., KEi) appear in a SECURITY PROTOCOL OUT parameter list. The responder is always the device server in this standard and all such IKE payloads (e.g., AUTHr) appear in SECURITY PROTOCOL IN parameter data.	

## C.2 ESP protocol details and variations for ESP-SCSI

The IKEv2 protocol details and variations specified in RFC 4303 apply to ESP-SCSI (i.e., this standard) as follows:

- This standard requires an integrity check value (icv field), whereas ESP allows support of confidentiality-only;
- This standard does not support traffic flow confidentiality;
- This standard does not support the TCP/IP aspects of ESP (e.g., IP addresses, multicast);
- This standard requires anti-replay detection using the sequence number, whereas ESP makes this optional;
- This standard does not support the Next Header field, but does reserve space for it in the MUST BE ZERO field (see table 77 in 5.14.7.3);
- This standard requires verification of the padding bytes, when possible;
- There is no provision in this standard for generating 'dummy packets'; and
- This standard does not support out-of-order parameter data.

## Annex D

(informative)

### Numeric order codes

#### D.1 Numeric order codes introduction

This annex contains SCSI additional sense codes, operation codes, diagnostic page codes, log page codes, mode page codes, VPD page codes, version descriptor values, and T10 IEEE binary identifiers in numeric order as a reference. In the event of a conflict with between the codes or usage requirements in this annex and equivalent information in the body of this standard or in any command standard, the normative codes and usage information is correct.

The information in this annex was complete and accurate at the time of publication. However, the information is subject to change. Technical Committee T10 of INCITS maintains an electronic copy of this information on its world wide web site (<http://www.t10.org/>). In the event that the T10 world wide web site is no longer active, access may be possible via the INCITS world wide web site (<http://www.incits.org/>), the ANSI world wide web site (<http://www.ansi.org/>), the IEC site (<http://www.iec.ch/>), the ISO site (<http://www.iso.ch/>), or the ISO/IEC JTC 1 web site (<http://www.jtc1.org/>).

#### D.2 Additional sense codes

Table D.1 is a numerical order listing of the additional sense codes (i.e., the ADDITIONAL SENSE CODE field and ADDITIONAL SENSE CODE QUALIFIER field values returned in sense data).

**Table D.1 — ASC and ASCQ assignments (part 1 of 17)**

D – Direct Access Block Device (SBC-3)			<u>Device Column key</u>
. T – Sequential Access Device (SSC-3)			blank = code not used
. L – Printer Device (SSC)			not blank = code used
. P – Processor Device (SPC-2)			
. W – Write Once Block Device (SBC)			
. R – C/DVD Device (MMC-6)			
. O – Optical Memory Block Device (SBC)			
. M – Media Changer Device (SMC-3)			
. A – Storage Array Device (SCC-2)			
. E – SCSI Enclosure Services device (SES)			
. B – Simplified Direct-Access (Reduced Block) device (RBC)			
. K – Optical Card Reader/Writer device (OCRW)			
. V – Automation/Device Interface device (ADC)			
. F – Object-based Storage Device (OSD)			
. . . . .			
ASC	ASCQ	DTLPWROMAEBKVF	Description
00h	00h	DTLPWROMAEBKVF	NO ADDITIONAL SENSE INFORMATION
00h	01h	T	FILEMARK DETECTED
00h	02h	T	END-OF-PARTITION/MEDIUM DETECTED
00h	03h	T	SETMARK DETECTED
00h	04h	T	BEGINNING-OF-PARTITION/MEDIUM DETECTED
00h	05h	TL	END-OF-DATA DETECTED
00h	06h	DTLPWROMAEBKVF	I/O PROCESS TERMINATED
00h	07h	T	PROGRAMMABLE EARLY WARNING DETECTED
00h	11h	R	AUDIO PLAY OPERATION IN PROGRESS
00h	12h	R	AUDIO PLAY OPERATION PAUSED
00h	13h	R	AUDIO PLAY OPERATION SUCCESSFULLY COMPLETED
00h	14h	R	AUDIO PLAY OPERATION STOPPED DUE TO ERROR

Table D.1 — ASC and ASCQ assignments (part 2 of 17)

D – Direct Access Block Device (SBC-3)										Device Column key
. T – Sequential Access Device (SSC-3)										blank = code not used
. L – Printer Device (SSC)										not blank = code used
. P – Processor Device (SPC-2)										
. . W – Write Once Block Device (SBC)										
. . R – C/DVD Device (MMC-6)										
. . O – Optical Memory Block Device (SBC)										
. . . M – Media Changer Device (SMC-3)										
. . . A – Storage Array Device (SCC-2)										
. . . E – SCSI Enclosure Services device (SES)										
. . . B – Simplified Direct-Access (Reduced Block) device (RBC)										
. . . K – Optical Card Reader/Writer device (OCRW)										
. . . V – Automation/Device Interface device (ADC)										
. . . F – Object-based Storage Device (OSD)										
ASC	ASCQ	DTLPWROMAEBKVF	Description							
00h	15h	R	NO CURRENT AUDIO STATUS TO RETURN							
00h	16h	DTLPWROMAEBKVF	OPERATION IN PROGRESS							
00h	17h	DTL WROMAEBKVF	CLEANING REQUESTED							
00h	18h	T	ERASE OPERATION IN PROGRESS							
00h	19h	T	LOCATE OPERATION IN PROGRESS							
00h	1Ah	T	REWIND OPERATION IN PROGRESS							
00h	1Bh	T	SET CAPACITY OPERATION IN PROGRESS							
00h	1Ch	T	VERIFY OPERATION IN PROGRESS							
00h	1Dh	DT B	ATA PASS THROUGH INFORMATION AVAILABLE							
00h	1Eh	DT R MAEBKV	CONFLICTING SA CREATION REQUEST							
01h	00h	D W O BK	NO INDEX/SECTOR SIGNAL							
02h	00h	D WRO BK	NO SEEK COMPLETE							
03h	00h	DTL W O BK	PERIPHERAL DEVICE WRITE FAULT							
03h	01h	T	NO WRITE CURRENT							
03h	02h	T	EXCESSIVE WRITE ERRORS							
04h	00h	DTLPWROMAEBKVF	LOGICAL UNIT NOT READY, CAUSE NOT REPORTABLE							
04h	01h	DTLPWROMAEBKVF	LOGICAL UNIT IS IN PROCESS OF BECOMING READY							
04h	02h	DTLPWROMAEBKVF	LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED							
04h	03h	DTLPWROMAEBKVF	LOGICAL UNIT NOT READY, MANUAL INTERVENTION REQUIRED							
04h	04h	DTL RO B	LOGICAL UNIT NOT READY, FORMAT IN PROGRESS							
04h	05h	DT W O A BK F	LOGICAL UNIT NOT READY, REBUILD IN PROGRESS							
04h	06h	DT W O A BK	LOGICAL UNIT NOT READY, RECALCULATION IN PROGRESS							
04h	07h	DTLPWROMAEBKVF	LOGICAL UNIT NOT READY, OPERATION IN PROGRESS							
04h	08h	R	LOGICAL UNIT NOT READY, LONG WRITE IN PROGRESS							
04h	09h	DTLPWROMAEBKVF	LOGICAL UNIT NOT READY, SELF-TEST IN PROGRESS							
04h	0Ah	DTLPWROMAEBKVF	LOGICAL UNIT NOT ACCESSIBLE, ASYMMETRIC ACCESS STATE TRANSITION							
04h	0Bh	DTLPWROMAEBKVF	LOGICAL UNIT NOT ACCESSIBLE, TARGET PORT IN STANDBY STATE							
04h	0Ch	DTLPWROMAEBKVF	LOGICAL UNIT NOT ACCESSIBLE, TARGET PORT IN UNAVAILABLE STATE							
04h	0Dh	F	LOGICAL UNIT NOT READY, STRUCTURE CHECK REQUIRED							
04h	10h	DT WROM B	LOGICAL UNIT NOT READY, AUXILIARY MEMORY NOT ACCESSIBLE							
04h	11h	DT WRO AEB VF	LOGICAL UNIT NOT READY, NOTIFY (ENABLE SPINUP) REQUIRED							
04h	12h	M V	LOGICAL UNIT NOT READY, OFFLINE							
04h	13h	DT R MAEBKV	LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS							
05h	00h	DTL WROMAEBKVF	LOGICAL UNIT DOES NOT RESPOND TO SELECTION							
06h	00h	D WROM BK	NO REFERENCE POSITION FOUND							
07h	00h	DTL WROM BK	MULTIPLE PERIPHERAL DEVICES SELECTED							
08h	00h	DTL WROMAEBKVF	LOGICAL UNIT COMMUNICATION FAILURE							
08h	01h	DTL WROMAEBKVF	LOGICAL UNIT COMMUNICATION TIME-OUT							
08h	02h	DTL WROMAEBKVF	LOGICAL UNIT COMMUNICATION PARITY ERROR							
08h	03h	DT ROM BK	LOGICAL UNIT COMMUNICATION CRC ERROR (ULTRA-DMA/32)							

Table D.1 — ASC and ASCQ assignments (part 3 of 17)

D – Direct Access Block Device (SBC-3)					Device Column key		
. T – Sequential Access Device (SSC-3)					blank = code not used		
. L – Printer Device (SSC)					not blank = code used		
. P – Processor Device (SPC-2)							
. W – Write Once Block Device (SBC)							
. R – C/DVD Device (MMC-6)							
. O – Optical Memory Block Device (SBC)							
. M – Media Changer Device (SMC-3)							
. A – Storage Array Device (SCC-2)							
. E – SCSI Enclosure Services device (SES)							
. B – Simplified Direct-Access (Reduced Block) device (RBC)							
. K – Optical Card Reader/Writer device (OCRW)							
. V – Automation/Device Interface device (ADC)							
. F – Object-based Storage Device (OSD)							
ASC	ASCQ	DTLPWROMAEBKVF	Description				
08h	04h	DTLPWRO	K	UNREACHABLE COPY TARGET			
09h	00h	DT	WRO	B	TRACK FOLLOWING ERROR		
09h	01h		WRO	K	TRACKING SERVO FAILURE		
09h	02h		WRO	K	FOCUS SERVO FAILURE		
09h	03h		WRO		SPINDLE SERVO FAILURE		
09h	04h	DT	WRO	B	HEAD SELECT FAULT		
0Ah	00h	DTLPWROMAEBKVF	ERROR LOG OVERFLOW				
0Bh	00h	DTLPWROMAEBKVF	WARNING				
0Bh	01h	DTLPWROMAEBKVF	WARNING - SPECIFIED TEMPERATURE EXCEEDED				
0Bh	02h	DTLPWROMAEBKVF	WARNING - ENCLOSURE DEGRADED				
0Bh	03h	DTLPWROMAEBKVF	WARNING - BACKGROUND SELF-TEST FAILED				
0Bh	04h	DTLPWRO	AEBKVF	WARNING - BACKGROUND PRE-SCAN DETECTED MEDIUM ERROR			
0Bh	05h	DTLPWRO	AEBKVF	WARNING - BACKGROUND MEDIUM SCAN DETECTED MEDIUM ERROR			
0Bh	06h	DTLPWROMAEBKVF	WARNING - NON-VOLATILE CACHE NOW VOLATILE				
0Bh	07h	DTLPWROMAEBKVF	WARNING - DEGRADED POWER TO NON-VOLATILE CACHE				
0Ch	00h	T	R	WRITE ERROR			
0Ch	01h			K	WRITE ERROR - RECOVERED WITH AUTO REALLOCATION		
0Ch	02h	D	W	O	BK	WRITE ERROR - AUTO REALLOCATION FAILED	
0Ch	03h	D	W	O	BK	WRITE ERROR - RECOMMEND REASSIGNMENT	
0Ch	04h	DT	W	O	B	COMPRESSION CHECK MISCOMPARE ERROR	
0Ch	05h	DT	W	O	B	DATA EXPANSION OCCURRED DURING COMPRESSION	
0Ch	06h	DT	W	O	B	BLOCK NOT COMPRESSIBLE	
0Ch	07h		R	WRITE ERROR - RECOVERY NEEDED			
0Ch	08h		R	WRITE ERROR - RECOVERY FAILED			
0Ch	09h		R	WRITE ERROR - LOSS OF STREAMING			
0Ch	0Ah		R	WRITE ERROR - PADDING BLOCKS ADDED			
0Ch	0Bh	DT	WROM	B	AUXILIARY MEMORY WRITE ERROR		
0Ch	0Ch	DTLPWRO	AEBKVF	WRITE ERROR - UNEXPECTED UNSOLICITED DATA			
0Ch	0Dh	DTLPWRO	AEBKVF	WRITE ERROR - NOT ENOUGH UNSOLICITED DATA			
0Ch	0Fh		R	DEFECTS IN ERROR WINDOW			
0Dh	00h	DTLPWRO	A	K	ERROR DETECTED BY THIRD PARTY TEMPORARY INITIATOR		
0Dh	01h	DTLPWRO	A	K	THIRD PARTY DEVICE FAILURE		
0Dh	02h	DTLPWRO	A	K	COPY TARGET DEVICE NOT REACHABLE		
0Dh	03h	DTLPWRO	A	K	INCORRECT COPY TARGET DEVICE TYPE		
0Dh	04h	DTLPWRO	A	K	COPY TARGET DEVICE DATA UNDERRUN		
0Dh	05h	DTLPWRO	A	K	COPY TARGET DEVICE DATA OVERRUN		
0Eh	00h	DT	PWROMAEBK	F	INVALID INFORMATION UNIT		
0Eh	01h	DT	PWROMAEBK	F	INFORMATION UNIT TOO SHORT		
0Eh	02h	DT	PWROMAEBK	F	INFORMATION UNIT TOO LONG		
0Eh	03h	DT	P	R	MAEBK	F	INVALID FIELD IN COMMAND INFORMATION UNIT
0Fh	00h						

Table D.1 — ASC and ASCQ assignments (part 4 of 17)

D – Direct Access Block Device (SBC-3)													Device Column key		
. T – Sequential Access Device (SSC-3)													blank = code not used		
. L – Printer Device (SSC)													not blank = code used		
. P – Processor Device (SPC-2)															
. W – Write Once Block Device (SBC)															
. R – C/DVD Device (MMC-6)															
. O – Optical Memory Block Device (SBC)															
. M – Media Changer Device (SMC-3)															
. A – Storage Array Device (SCC-2)															
. E – SCSI Enclosure Services device (SES)															
. B – Simplified Direct-Access (Reduced Block) device (RBC)															
. K – Optical Card Reader/Writer device (OCRW)															
. V – Automation/Device Interface device (ADC)															
. F – Object-based Storage Device (OSD)															
ASC	ASCQ	DT	L	P	W	R	O	M	A	E	B	K	V	F	Description
10h	00h	D			W	O					BK				ID CRC OR ECC ERROR
10h	01h	DT			W	O									LOGICAL BLOCK GUARD CHECK FAILED
10h	02h	DT			W	O									LOGICAL BLOCK APPLICATION TAG CHECK FAILED
10h	03h	DT			W	O									LOGICAL BLOCK REFERENCE TAG CHECK FAILED
11h	00h	DT			W	R	O				BK				UNRECOVERED READ ERROR
11h	01h	DT			W	R	O				BK				READ RETRIES EXHAUSTED
11h	02h	DT			W	R	O				BK				ERROR TOO LONG TO CORRECT
11h	03h	DT			W	O					BK				MULTIPLE READ ERRORS
11h	04h	D			W	O					BK				UNRECOVERED READ ERROR - AUTO REALLOCATE FAILED
11h	05h					W	R	O			B				L-EC UNCORRECTABLE ERROR
11h	06h					W	R	O			B				CIRC UNRECOVERED ERROR
11h	07h					W	O				B				DATA RE-SYNCHRONIZATION ERROR
11h	08h		T												INCOMPLETE BLOCK READ
11h	09h		T												NO GAP FOUND
11h	0Ah	DT					O				BK				MISCORRECTED ERROR
11h	0Bh	D			W	O					BK				UNRECOVERED READ ERROR - RECOMMEND REASSIGNMENT
11h	0Ch	D			W	O					BK				UNRECOVERED READ ERROR - RECOMMEND REWRITE THE DATA
11h	0Dh	DT			W	R	O				B				DE-COMPRESSION CRC ERROR
11h	0Eh	DT			W	R	O				B				CANNOT DECOMPRESS USING DECLARED ALGORITHM
11h	0Fh					R									ERROR READING UPC/EAN NUMBER
11h	10h					R									ERROR READING ISRC NUMBER
11h	11h					R									READ ERROR - LOSS OF STREAMING
11h	12h	DT			W	R	O	M			B				AUXILIARY MEMORY READ ERROR
11h	13h	DT	L	P	W	R	O		A	E	B	K	V	F	READ ERROR - FAILED RETRANSMISSION REQUEST
11h	14h	D													READ ERROR - LBA MARKED BAD BY APPLICATION CLIENT
12h	00h	D			W	O					BK				ADDRESS MARK NOT FOUND FOR ID FIELD
13h	00h	D			W	O					BK				ADDRESS MARK NOT FOUND FOR DATA FIELD
14h	00h	DT	L		W	R	O				BK				RECORDED ENTITY NOT FOUND
14h	01h	DT			W	R	O				BK				RECORD NOT FOUND
14h	02h		T												FILEMARK OR SETMARK NOT FOUND
14h	03h		T												END-OF-DATA NOT FOUND
14h	04h		T												BLOCK SEQUENCE ERROR
14h	05h	DT			W	O					BK				RECORD NOT FOUND - RECOMMEND REASSIGNMENT
14h	06h	DT			W	O					BK				RECORD NOT FOUND - DATA AUTO-REALLOCATED
14h	07h		T												LOCATE OPERATION FAILURE
15h	00h	DT	L		W	R	O	M			BK				RANDOM POSITIONING ERROR
15h	01h	DT	L		W	R	O	M			BK				MECHANICAL POSITIONING ERROR
15h	02h	DT			W	R	O				BK				POSITIONING ERROR DETECTED BY READ OF MEDIUM
16h	00h	D			W	O					BK				DATA SYNCHRONIZATION MARK ERROR
16h	01h	D			W	O					BK				DATA SYNC ERROR - DATA REWRITTEN
16h	02h	D			W	O					BK				DATA SYNC ERROR - RECOMMEND REWRITE

Table D.1 — ASC and ASCQ assignments (part 5 of 17)

D – Direct Access Block Device (SBC-3)														Device Column key	
. T – Sequential Access Device (SSC-3)														blank = code not used	
. L – Printer Device (SSC)														not blank = code used	
. P – Processor Device (SPC-2)															
. W – Write Once Block Device (SBC)															
. R – C/DVD Device (MMC-6)															
. O – Optical Memory Block Device (SBC)															
. M – Media Changer Device (SMC-3)															
. A – Storage Array Device (SCC-2)															
. E – SCSI Enclosure Services device (SES)															
. B – Simplified Direct-Access (Reduced Block) device (RBC)															
. K – Optical Card Reader/Writer device (OCRW)															
. V – Automation/Device Interface device (ADC)															
. F – Object-based Storage Device (OSD)															
ASC	ASCQ	DT	L	P	W	R	O	M	A	E	B	K	V	F	Description
16h	03h	D			W		O					BK			DATA SYNC ERROR - DATA AUTO-REALLOCATED
16h	04h	D			W		O					BK			DATA SYNC ERROR - RECOMMEND REASSIGNMENT
17h	00h	DT			W		R	O				BK			RECOVERED DATA WITH NO ERROR CORRECTION APPLIED
17h	01h	DT			W		R	O				BK			RECOVERED DATA WITH RETRIES
17h	02h	DT			W		R	O				BK			RECOVERED DATA WITH POSITIVE HEAD OFFSET
17h	03h	DT			W		R	O				BK			RECOVERED DATA WITH NEGATIVE HEAD OFFSET
17h	04h				W		R	O				B			RECOVERED DATA WITH RETRIES AND/OR CIRC APPLIED
17h	05h	D			W		R	O				BK			RECOVERED DATA USING PREVIOUS SECTOR ID
17h	06h	D			W		O					BK			RECOVERED DATA WITHOUT ECC - DATA AUTO-REALLOCATED
17h	07h	D			W		R	O				BK			RECOVERED DATA WITHOUT ECC - RECOMMEND REASSIGNMENT
17h	08h	D			W		R	O				BK			RECOVERED DATA WITHOUT ECC - RECOMMEND REWRITE
17h	09h	D			W		R	O				BK			RECOVERED DATA WITHOUT ECC - DATA REWRITTEN
18h	00h	DT			W		R	O				BK			RECOVERED DATA WITH ERROR CORRECTION APPLIED
18h	01h	D			W		R	O				BK			RECOVERED DATA WITH ERROR CORR. & RETRIES APPLIED
18h	02h	D			W		R	O				BK			RECOVERED DATA - DATA AUTO-REALLOCATED
18h	03h						R								RECOVERED DATA WITH CIRC
18h	04h						R								RECOVERED DATA WITH L-EC
18h	05h	D			W		R	O				BK			RECOVERED DATA - RECOMMEND REASSIGNMENT
18h	06h	D			W		R	O				BK			RECOVERED DATA - RECOMMEND REWRITE
18h	07h	D			W		O					BK			RECOVERED DATA WITH ECC - DATA REWRITTEN
18h	08h						R								RECOVERED DATA WITH LINKING
19h	00h	D					O					K			DEFECT LIST ERROR
19h	01h	D					O					K			DEFECT LIST NOT AVAILABLE
19h	02h	D					O					K			DEFECT LIST ERROR IN PRIMARY LIST
19h	03h	D					O					K			DEFECT LIST ERROR IN GROWN LIST
1Ah	00h	DTLPW			R	O		M	A	E	B	K	V	F	PARAMETER LIST LENGTH ERROR
1Bh	00h	DTLPW			R	O		M	A	E	B	K	V	F	SYNCHRONOUS DATA TRANSFER ERROR
1Ch	00h	D					O					BK			DEFECT LIST NOT FOUND
1Ch	01h	D					O					BK			PRIMARY DEFECT LIST NOT FOUND
1Ch	02h	D					O					BK			GROWN DEFECT LIST NOT FOUND
1Dh	00h	DT			W		R	O				BK			MISCOMPARE DURING VERIFY OPERATION
1Eh	00h	D			W		O					BK			RECOVERED ID WITH ECC CORRECTION
1Fh	00h	D					O					K			PARTIAL DEFECT LIST TRANSFER
20h	00h	DTLPW			R	O		M	A	E	B	K	V	F	INVALID COMMAND OPERATION CODE
20h	01h	DT			P		W	R	O		M	A	E	B	ACCESS DENIED - INITIATOR PENDING-ENROLLED
20h	02h	DT			P		W	R	O		M	A	E	B	ACCESS DENIED - NO ACCESS RIGHTS
20h	03h	DT			P		W	R	O		M	A	E	B	ACCESS DENIED - INVALID MGMT ID KEY
20h	04h	T													ILLEGAL COMMAND WHILE IN WRITE CAPABLE STATE
20h	05h	T													Obsolete
20h	06h	T													ILLEGAL COMMAND WHILE IN EXPLICIT ADDRESS MODE
20h	07h	T													ILLEGAL COMMAND WHILE IN IMPLICIT ADDRESS MODE



Table D.1 — ASC and ASCQ assignments (part 6 of 17)

D – Direct Access Block Device (SBC-3)				Device Column key
. T – Sequential Access Device (SSC-3)				blank = code not used
. L – Printer Device (SSC)				not blank = code used
. P – Processor Device (SPC-2)				
. W – Write Once Block Device (SBC)				
. R – C/DVD Device (MMC-6)				
. O – Optical Memory Block Device (SBC)				
. M – Media Changer Device (SMC-3)				
. A – Storage Array Device (SCC-2)				
. E – SCSI Enclosure Services device (SES)				
. B – Simplified Direct-Access (Reduced Block) device (RBC)				
. K – Optical Card Reader/Writer device (OCRW)				
. V – Automation/Device Interface device (ADC)				
. F – Object-based Storage Device (OSD)				
ASC	ASCQ	DTLPWROMAEBKVF	Description	
20h	08h	DT PWROMAEBK	ACCESS DENIED - ENROLLMENT CONFLICT	
20h	09h	DT PWROMAEBK	ACCESS DENIED - INVALID LU IDENTIFIER	
20h	0Ah	DT PWROMAEBK	ACCESS DENIED - INVALID PROXY TOKEN	
20h	0Bh	DT PWROMAEBK	ACCESS DENIED - ACL LUN CONFLICT	
21h	00h	DT WRO BK	LOGICAL BLOCK ADDRESS OUT OF RANGE	
21h	01h	DT WROM BK	INVALID ELEMENT ADDRESS	
21h	02h	R	INVALID ADDRESS FOR WRITE	
21h	03h	R	INVALID WRITE CROSSING LAYER JUMP	
22h	00h	D	ILLEGAL FUNCTION (USE 20 00, 24 00, OR 26 00)	
23h	00h			
24h	00h	DTLPWROMAEBKVF	INVALID FIELD IN CDB	
24h	01h	DTLPWRO AEBKVF	CDB DECRYPTION ERROR	
24h	02h	T	Obsolete	
24h	03h	T	Obsolete	
24h	04h		F	SECURITY AUDIT VALUE FROZEN
24h	05h		F	SECURITY WORKING KEY FROZEN
24h	06h		F	NONCE NOT UNIQUE
24h	07h		F	NONCE TIMESTAMP OUT OF RANGE
24h	08h	DT R MAEBKV	INVALID XCDB	
25h	00h	DTLPWROMAEBKVF	LOGICAL UNIT NOT SUPPORTED	
26h	00h	DTLPWROMAEBKVF	INVALID FIELD IN PARAMETER LIST	
26h	01h	DTLPWROMAEBKVF	PARAMETER NOT SUPPORTED	
26h	02h	DTLPWROMAEBKVF	PARAMETER VALUE INVALID	
26h	03h	DTLPWROMAE K	THRESHOLD PARAMETERS NOT SUPPORTED	
26h	04h	DTLPWROMAEBKVF	INVALID RELEASE OF PERSISTENT RESERVATION	
26h	05h	DTLPWRO A BK	DATA DECRYPTION ERROR	
26h	06h	DTLPWRO K	TOO MANY TARGET DESCRIPTORS	
26h	07h	DTLPWRO K	UNSUPPORTED TARGET DESCRIPTOR TYPE CODE	
26h	08h	DTLPWRO K	TOO MANY SEGMENT DESCRIPTORS	
26h	09h	DTLPWRO K	UNSUPPORTED SEGMENT DESCRIPTOR TYPE CODE	
26h	0Ah	DTLPWRO K	UNEXPECTED INEXACT SEGMENT	
26h	0Bh	DTLPWRO K	INLINE DATA LENGTH EXCEEDED	
26h	0Ch	DTLPWRO K	INVALID OPERATION FOR COPY SOURCE OR DESTINATION	
26h	0Dh	DTLPWRO K	COPY SEGMENT GRANULARITY VIOLATION	
26h	0Eh	DT PWROMAEBK	INVALID PARAMETER WHILE PORT IS ENABLED	
26h	0Fh		F	INVALID DATA-OUT BUFFER INTEGRITY CHECK VALUE
26h	10h	T	DATA DECRYPTION KEY FAIL LIMIT REACHED	
26h	11h	T	INCOMPLETE KEY-ASSOCIATED DATA SET	
26h	12h	T	VENDOR SPECIFIC KEY REFERENCE NOT FOUND	
27h	00h	DT WRO BK	WRITE PROTECTED	
27h	01h	DT WRO BK	HARDWARE WRITE PROTECTED	

Table D.1 — ASC and ASCQ assignments (part 7 of 17)

D – Direct Access Block Device (SBC-3)					Device Column key
. T – Sequential Access Device (SSC-3)					blank = code not used
. L – Printer Device (SSC)					not blank = code used
. P – Processor Device (SPC-2)					
. W – Write Once Block Device (SBC)					
. R – C/DVD Device (MMC-6)					
. O – Optical Memory Block Device (SBC)					
. M – Media Changer Device (SMC-3)					
. A – Storage Array Device (SCC-2)					
. E – SCSI Enclosure Services device (SES)					
. B – Simplified Direct-Access (Reduced Block) device (RBC)					
. K – Optical Card Reader/Writer device (OCRW)					
. V – Automation/Device Interface device (ADC)					
. F – Object-based Storage Device (OSD)					
ASC	ASCQ	DTLPWROMAEBKVF	Description		
27h	02h	DT WRO BK	LOGICAL UNIT SOFTWARE WRITE PROTECTED		
27h	03h	T R	ASSOCIATED WRITE PROTECT		
27h	04h	T R	PERSISTENT WRITE PROTECT		
27h	05h	T R	PERMANENT WRITE PROTECT		
27h	06h	R F	CONDITIONAL WRITE PROTECT		
28h	00h	DTLPWROMAEBKVF	NOT READY TO READY CHANGE, MEDIUM MAY HAVE CHANGED		
28h	01h	DT WROM B	IMPORT OR EXPORT ELEMENT ACCESSED		
28h	02h	R	FORMAT-LAYER MAY HAVE CHANGED		
28h	03h	M	IMPORT/EXPORT ELEMENT ACCESSED, MEDIUM CHANGED		
29h	00h	DTLPWROMAEBKVF	POWER ON, RESET, OR BUS DEVICE RESET OCCURRED		
29h	01h	DTLPWROMAEBKVF	POWER ON OCCURRED		
29h	02h	DTLPWROMAEBKVF	SCSI BUS RESET OCCURRED		
29h	03h	DTLPWROMAEBKVF	BUS DEVICE RESET FUNCTION OCCURRED		
29h	04h	DTLPWROMAEBKVF	DEVICE INTERNAL RESET		
29h	05h	DTLPWROMAEBKVF	TRANSCEIVER MODE CHANGED TO SINGLE-ENDED		
29h	06h	DTLPWROMAEBKVF	TRANSCEIVER MODE CHANGED TO LVD		
29h	07h	DTLPWROMAEBKVF	I_T NEXUS LOSS OCCURRED		
2Ah	00h	DTL WROMAEBKVF	PARAMETERS CHANGED		
2Ah	01h	DTL WROMAEBKVF	MODE PARAMETERS CHANGED		
2Ah	02h	DTL WROMAE K	LOG PARAMETERS CHANGED		
2Ah	03h	DTLPWROMAE K	RESERVATIONS PREEMPTED		
2Ah	04h	DTLPWROMAE	RESERVATIONS RELEASED		
2Ah	05h	DTLPWROMAE	REGISTRATIONS PREEMPTED		
2Ah	06h	DTLPWROMAEBKVF	ASYMMETRIC ACCESS STATE CHANGED		
2Ah	07h	DTLPWROMAEBKVF	IMPLICIT ASYMMETRIC ACCESS STATE TRANSITION FAILED		
2Ah	08h	DT WROMAEBKVF	PRIORITY CHANGED		
2Ah	09h	D	CAPACITY DATA HAS CHANGED		
2Ah	0Ah	DT	ERROR HISTORY I_T NEXUS CLEARED		
2Ah	0Bh	DT	ERROR HISTORY SNAPSHOT RELEASED		
2Ah	0Ch	F	ERROR RECOVERY ATTRIBUTES HAVE CHANGED		
2Ah	0Dh	T	DATA ENCRYPTION CAPABILITIES CHANGED		
2Ah	10h	DT M E V	TIMESTAMP CHANGED		
2Ah	11h	T	DATA ENCRYPTION PARAMETERS CHANGED BY ANOTHER I_T NEXUS		
2Ah	12h	T	DATA ENCRYPTION PARAMETERS CHANGED BY VENDOR SPECIFIC EVENT		
2Ah	13h	T	DATA ENCRYPTION KEY INSTANCE COUNTER HAS CHANGED		
2Ah	14h	DT R MAEBKV	SA CREATION CAPABILITIES DATA HAS CHANGED		
2Bh	00h	DTLPWRO K	COPY CANNOT EXECUTE SINCE HOST CANNOT DISCONNECT		
2Ch	00h	DTLPWROMAEBKVF	COMMAND SEQUENCE ERROR		
2Ch	01h		TOO MANY WINDOWS SPECIFIED		
2Ch	02h		INVALID COMBINATION OF WINDOWS SPECIFIED		
2Ch	03h	R	CURRENT PROGRAM AREA IS NOT EMPTY		

Table D.1 — ASC and ASCQ assignments (part 8 of 17)

		D – Direct Access Block Device (SBC-3) . T – Sequential Access Device (SSC-3) . L – Printer Device (SSC) . P – Processor Device (SPC-2) . . W – Write Once Block Device (SBC) . . R – C/DVD Device (MMC-6) . . O – Optical Memory Block Device (SBC) . . . M – Media Changer Device (SMC-3) . . . A – Storage Array Device (SCC-2) . . . E – SCSI Enclosure Services device (SES) . . . B – Simplified Direct-Access (Reduced Block) device (RBC) . . . K – Optical Card Reader/Writer device (OCRW) . . . V – Automation/Device Interface device (ADC) . . . F – Object-based Storage Device (OSD)					<u>Device Column key</u> blank = code not used not blank = code used
ASC	ASCQ	DTLPWROMAEBKVF	Description				
2Ch	04h		R				CURRENT PROGRAM AREA IS EMPTY
2Ch	05h				B		ILLEGAL POWER CONDITION REQUEST
2Ch	06h		R				PERSISTENT PREVENT CONFLICT
2Ch	07h	DTLPWROMAEBKVF					PREVIOUS BUSY STATUS
2Ch	08h	DTLPWROMAEBKVF					PREVIOUS TASK SET FULL STATUS
2Ch	09h	DTLPWROM		EBKVF			PREVIOUS RESERVATION CONFLICT STATUS
2Ch	0Ah				F		PARTITION OR COLLECTION CONTAINS USER OBJECTS
2Ch	0Bh	T					NOT RESERVED
2Dh	00h	T					OVERWRITE ERROR ON UPDATE IN PLACE
2Eh	00h		R				INSUFFICIENT TIME FOR OPERATION
2Fh	00h	DTLPWROMAEBKVF					COMMANDS CLEARED BY ANOTHER INITIATOR
2Fh	01h	D					COMMANDS CLEARED BY POWER LOSS NOTIFICATION
2Fh	02h	DTLPWROMAEBKVF					COMMANDS CLEARED BY DEVICE SERVER
30h	00h	DT	WROM		BK		INCOMPATIBLE MEDIUM INSTALLED
30h	01h	DT	WRO		BK		CANNOT READ MEDIUM - UNKNOWN FORMAT
30h	02h	DT	WRO		BK		CANNOT READ MEDIUM - INCOMPATIBLE FORMAT
30h	03h	DT	R		K		CLEANING CARTRIDGE INSTALLED
30h	04h	DT	WRO		BK		CANNOT WRITE MEDIUM - UNKNOWN FORMAT
30h	05h	DT	WRO		BK		CANNOT WRITE MEDIUM - INCOMPATIBLE FORMAT
30h	06h	DT	WRO		B		CANNOT FORMAT MEDIUM - INCOMPATIBLE MEDIUM
30h	07h	DTL	WROMAEBKVF				CLEANING FAILURE
30h	08h		R				CANNOT WRITE - APPLICATION CODE MISMATCH
30h	09h		R				CURRENT SESSION NOT FIXATED FOR APPEND
30h	0Ah	DT	WRO	AEBK			CLEANING REQUEST REJECTED
30h	0Ch	T					WORM MEDIUM - OVERWRITE ATTEMPTED
30h	0Dh	T					WORM MEDIUM - INTEGRITY CHECK
30h	10h		R				MEDIUM NOT FORMATTED
30h	11h			M			INCOMPATIBLE VOLUME TYPE
30h	12h			M			INCOMPATIBLE VOLUME QUALIFIER
31h	00h	DT	WRO		BK		MEDIUM FORMAT CORRUPTED
31h	01h	D L	RO		B		FORMAT COMMAND FAILED
31h	02h		R				ZONED FORMATTING FAILED DUE TO SPARE LINKING
32h	00h	D	W O		BK		NO DEFECT SPARE LOCATION AVAILABLE
32h	01h	D	W O		BK		DEFECT LIST UPDATE FAILURE
33h	00h	T					TAPE LENGTH ERROR
34h	00h	DTLPWROMAEBKVF					ENCLOSURE FAILURE
35h	00h	DTLPWROMAEBKVF					ENCLOSURE SERVICES FAILURE
35h	01h	DTLPWROMAEBKVF					UNSUPPORTED ENCLOSURE FUNCTION
35h	02h	DTLPWROMAEBKVF					ENCLOSURE SERVICES UNAVAILABLE
35h	03h	DTLPWROMAEBKVF					ENCLOSURE SERVICES TRANSFER FAILURE
35h	04h	DTLPWROMAEBKVF					ENCLOSURE SERVICES TRANSFER REFUSED

Table D.1 — ASC and ASCQ assignments (part 9 of 17)

		D – Direct Access Block Device (SBC-3) . T – Sequential Access Device (SSC-3) . L – Printer Device (SSC) . P – Processor Device (SPC-2) . . W – Write Once Block Device (SBC) . . R – C/DVD Device (MMC-6) . . O – Optical Memory Block Device (SBC) . . . M – Media Changer Device (SMC-3) . . . A – Storage Array Device (SCC-2) . . . E – SCSI Enclosure Services device (SES) . . . B – Simplified Direct-Access (Reduced Block) device (RBC) . . . K – Optical Card Reader/Writer device (OCRW) . . . V – Automation/Device Interface device (ADC) . . . F – Object-based Storage Device (OSD)						Device Column key blank = code not used not blank = code used
ASC	ASCQ	DTLPWROMAEBKVF	Description					
35h	05h	DTL	WROMAEBKVF					ENCLOSURE SERVICES CHECKSUM ERROR
36h	00h		L					RIBBON, INK, OR TONER FAILURE
37h	00h	DTL	WROMAEBKVF					ROUNDED PARAMETER
38h	00h					B		EVENT STATUS NOTIFICATION
38h	02h					B		ESN - POWER MANAGEMENT CLASS EVENT
38h	04h					B		ESN - MEDIA CLASS EVENT
38h	06h					B		ESN - DEVICE BUSY CLASS EVENT
39h	00h	DTL	WROMAE			K		SAVING PARAMETERS NOT SUPPORTED
3Ah	00h	DTL	WROM			BK		MEDIUM NOT PRESENT
3Ah	01h	DT	WROM			BK		MEDIUM NOT PRESENT - TRAY CLOSED
3Ah	02h	DT	WROM			BK		MEDIUM NOT PRESENT - TRAY OPEN
3Ah	03h	DT	WROM			B		MEDIUM NOT PRESENT - LOADABLE
3Ah	04h	DT	WRO			B		MEDIUM NOT PRESENT - MEDIUM AUXILIARY MEMORY ACCESSIBLE
3Bh	00h		T	L				SEQUENTIAL POSITIONING ERROR
3Bh	01h		T					TAPE POSITION ERROR AT BEGINNING-OF-MEDIUM
3Bh	02h		T					TAPE POSITION ERROR AT END-OF-MEDIUM
3Bh	03h		L					TAPE OR ELECTRONIC VERTICAL FORMS UNIT NOT READY
3Bh	04h		L					SLEW FAILURE
3Bh	05h		L					PAPER JAM
3Bh	06h		L					FAILED TO SENSE TOP-OF-FORM
3Bh	07h		L					FAILED TO SENSE BOTTOM-OF-FORM
3Bh	08h		T					REPOSITION ERROR
3Bh	09h							READ PAST END OF MEDIUM
3Bh	0Ah							READ PAST BEGINNING OF MEDIUM
3Bh	0Bh							POSITION PAST END OF MEDIUM
3Bh	0Ch		T					POSITION PAST BEGINNING OF MEDIUM
3Bh	0Dh	DT	WROM			BK		MEDIUM DESTINATION ELEMENT FULL
3Bh	0Eh	DT	WROM			BK		MEDIUM SOURCE ELEMENT EMPTY
3Bh	0Fh		R					END OF MEDIUM REACHED
3Bh	11h	DT	WROM			BK		MEDIUM MAGAZINE NOT ACCESSIBLE
3Bh	12h	DT	WROM			BK		MEDIUM MAGAZINE REMOVED
3Bh	13h	DT	WROM			BK		MEDIUM MAGAZINE INSERTED
3Bh	14h	DT	WROM			BK		MEDIUM MAGAZINE LOCKED
3Bh	15h	DT	WROM			BK		MEDIUM MAGAZINE UNLOCKED
3Bh	16h		R					MECHANICAL POSITIONING OR CHANGER ERROR
3Bh	17h					F		READ PAST END OF USER OBJECT
3Bh	18h					M		ELEMENT DISABLED
3Bh	19h					M		ELEMENT ENABLED
3Bh	1Ah					M		DATA TRANSFER DEVICE REMOVED
3Bh	1Bh					M		DATA TRANSFER DEVICE INSERTED
3Ch	00h							

Table D.1 — ASC and ASCQ assignments (part 10 of 17)

D – Direct Access Block Device (SBC-3)			Device Column key
. T – Sequential Access Device (SSC-3)			blank = code not used
. L – Printer Device (SSC)			not blank = code used
. P – Processor Device (SPC-2)			
. W – Write Once Block Device (SBC)			
. R – C/DVD Device (MMC-6)			
. O – Optical Memory Block Device (SBC)			
. M – Media Changer Device (SMC-3)			
. A – Storage Array Device (SCC-2)			
. E – SCSI Enclosure Services device (SES)			
. B – Simplified Direct-Access (Reduced Block) device (RBC)			
. K – Optical Card Reader/Writer device (OCRW)			
. V – Automation/Device Interface device (ADC)			
. F – Object-based Storage Device (OSD)			
ASC	ASCQ	DTLPWROMAEBKVF	Description
3Dh	00h	DTLPWROMAE K	INVALID BITS IN IDENTIFY MESSAGE
3Eh	00h	DTLPWROMAEBKVF	LOGICAL UNIT HAS NOT SELF-CONFIGURED YET
3Eh	01h	DTLPWROMAEBKVF	LOGICAL UNIT FAILURE
3Eh	02h	DTLPWROMAEBKVF	TIMEOUT ON LOGICAL UNIT
3Eh	03h	DTLPWROMAEBKVF	LOGICAL UNIT FAILED SELF-TEST
3Eh	04h	DTLPWROMAEBKVF	LOGICAL UNIT UNABLE TO UPDATE SELF-TEST LOG
3Fh	00h	DTLPWROMAEBKVF	TARGET OPERATING CONDITIONS HAVE CHANGED
3Fh	01h	DTLPWROMAEBKVF	MICROCODE HAS BEEN CHANGED
3Fh	02h	DTLPWROM BK	CHANGED OPERATING DEFINITION
3Fh	03h	DTLPWROMAEBKVF	INQUIRY DATA HAS CHANGED
3Fh	04h	DT WROMAEBK	COMPONENT DEVICE ATTACHED
3Fh	05h	DT WROMAEBK	DEVICE IDENTIFIER CHANGED
3Fh	06h	DT WROMAEB	REDUNDANCY GROUP CREATED OR MODIFIED
3Fh	07h	DT WROMAEB	REDUNDANCY GROUP DELETED
3Fh	08h	DT WROMAEB	SPARE CREATED OR MODIFIED
3Fh	09h	DT WROMAEB	SPARE DELETED
3Fh	0Ah	DT WROMAEBK	VOLUME SET CREATED OR MODIFIED
3Fh	0Bh	DT WROMAEBK	VOLUME SET DELETED
3Fh	0Ch	DT WROMAEBK	VOLUME SET DEASSIGNED
3Fh	0Dh	DT WROMAEBK	VOLUME SET REASSIGNED
3Fh	0Eh	DTLPWROMAE	REPORTED LUNS DATA HAS CHANGED
3Fh	0Fh	DTLPWROMAEBKVF	ECHO BUFFER OVERWRITTEN
3Fh	10h	DT WROM B	MEDIUM LOADABLE
3Fh	11h	DT WROM B	MEDIUM AUXILIARY MEMORY ACCESSIBLE
3Fh	12h	DTLPWR MAEBK F	iSCSI IP ADDRESS ADDED
3Fh	13h	DTLPWR MAEBK F	iSCSI IP ADDRESS REMOVED
3Fh	14h	DTLPWR MAEBK F	iSCSI IP ADDRESS CHANGED
40h	00h	D	RAM FAILURE (SHOULD USE 40 NN)
40h	NNh	DTLPWROMAEBKVF	DIAGNOSTIC FAILURE ON COMPONENT NN (80h-FFh)
41h	00h	D	DATA PATH FAILURE (SHOULD USE 40 NN)
42h	00h	D	POWER-ON OR SELF-TEST FAILURE (SHOULD USE 40 NN)
43h	00h	DTLPWROMAEBKVF	MESSAGE ERROR
44h	00h	DTLPWROMAEBKVF	INTERNAL TARGET FAILURE
44h	71h	DT B	ATA DEVICE FAILED SET FEATURES
45h	00h	DTLPWROMAEBKVF	SELECT OR RESELECT FAILURE
46h	00h	DTLPWROM BK	UNSUCCESSFUL SOFT RESET
47h	00h	DTLPWROMAEBKVF	SCSI PARITY ERROR
47h	01h	DTLPWROMAEBKVF	DATA PHASE CRC ERROR DETECTED
47h	02h	DTLPWROMAEBKVF	SCSI PARITY ERROR DETECTED DURING ST DATA PHASE
47h	03h	DTLPWROMAEBKVF	INFORMATION UNIT iuCRC ERROR DETECTED
47h	04h	DTLPWROMAEBKVF	ASYNCHRONOUS INFORMATION PROTECTION ERROR DETECTED

**Table D.1 — ASC and ASCQ assignments** (part 11 of 17)

D – Direct Access Block Device (SBC-3)				Device Column key
. T – Sequential Access Device (SSC-3)				blank = code not used
. L – Printer Device (SSC)				not blank = code used
. P – Processor Device (SPC-2)				
. W – Write Once Block Device (SBC)				
. R – C/DVD Device (MMC-6)				
. O – Optical Memory Block Device (SBC)				
. M – Media Changer Device (SMC-3)				
. A – Storage Array Device (SCC-2)				
. E – SCSI Enclosure Services device (SES)				
. B – Simplified Direct-Access (Reduced Block) device (RBC)				
. K – Optical Card Reader/Writer device (OCRW)				
. V – Automation/Device Interface device (ADC)				
. F – Object-based Storage Device (OSD)				
ASC	ASCQ	DTLPWROMAEBKVF	Description	
47h	05h	DTLPWROMAEBKVF	PROTOCOL SERVICE CRC ERROR	
47h	06h	DT MAEBKVF	PHY TEST FUNCTION IN PROGRESS	
47h	7Fh	DT PWROMAEBK	SOME COMMANDS CLEARED BY ISCSI PROTOCOL EVENT	
48h	00h	DTLPWROMAEBKVF	INITIATOR DETECTED ERROR MESSAGE RECEIVED	
49h	00h	DTLPWROMAEBKVF	INVALID MESSAGE ERROR	
4Ah	00h	DTLPWROMAEBKVF	COMMAND PHASE ERROR	
4Bh	00h	DTLPWROMAEBKVF	DATA PHASE ERROR	
4Bh	01h	DT PWROMAEBK	INVALID TARGET PORT TRANSFER TAG RECEIVED	
4Bh	02h	DT PWROMAEBK	TOO MUCH WRITE DATA	
4Bh	03h	DT PWROMAEBK	ACK/NAK TIMEOUT	
4Bh	04h	DT PWROMAEBK	NAK RECEIVED	
4Bh	05h	DT PWROMAEBK	DATA OFFSET ERROR	
4Bh	06h	DT PWROMAEBK	INITIATOR RESPONSE TIMEOUT	
4Ch	00h	DTLPWROMAEBKVF	LOGICAL UNIT FAILED SELF-CONFIGURATION	
4Dh	NNh	DTLPWROMAEBKVF	TAGGED OVERLAPPED COMMANDS (NN = TASK TAG)	
4Eh	00h	DTLPWROMAEBKVF	OVERLAPPED COMMANDS ATTEMPTED	
4Fh	00h			
50h	00h	T	WRITE APPEND ERROR	
50h	01h	T	WRITE APPEND POSITION ERROR	
50h	02h	T	POSITION ERROR RELATED TO TIMING	
51h	00h	T RO	ERASE FAILURE	
51h	01h	R	ERASE FAILURE - INCOMPLETE ERASE OPERATION DETECTED	
52h	00h	T	CARTRIDGE FAULT	
53h	00h	DTL WROM BK	MEDIA LOAD OR EJECT FAILED	
53h	01h	T	UNLOAD TAPE FAILURE	
53h	02h	DT WROM BK	MEDIUM REMOVAL PREVENTED	
53h	03h	M	MEDIUM REMOVAL PREVENTED BY DATA TRANSFER ELEMENT	
53h	04h	T	MEDIUM THREAD OR UNTHREAD FAILURE	
54h	00h	P	SCSI TO HOST SYSTEM INTERFACE FAILURE	
55h	00h	P	SYSTEM RESOURCE FAILURE	
55h	01h	D O BK	SYSTEM BUFFER FULL	
55h	02h	DTLPWROMAE K	INSUFFICIENT RESERVATION RESOURCES	
55h	03h	DTLPWROMAE K	INSUFFICIENT RESOURCES	
55h	04h	DTLPWROMAE K	INSUFFICIENT REGISTRATION RESOURCES	
55h	05h	DT PWROMAEBK	INSUFFICIENT ACCESS CONTROL RESOURCES	
55h	06h	DT WROM B	AUXILIARY MEMORY OUT OF SPACE	
55h	07h		F QUOTA ERROR	
55h	08h	T	MAXIMUM NUMBER OF SUPPLEMENTAL DECRYPTION KEYS EXCEEDED	
55h	09h	M	MEDIUM AUXILIARY MEMORY NOT ACCESSIBLE	
55h	0Ah	M	DATA CURRENTLY UNAVAILABLE	
56h	00h			

Table D.1 — ASC and ASCQ assignments (part 12 of 17)

D – Direct Access Block Device (SBC-3)					Device Column key	
. T – Sequential Access Device (SSC-3)					blank = code not used	
. L – Printer Device (SSC)					not blank = code used	
. P – Processor Device (SPC-2)						
. . W – Write Once Block Device (SBC)						
. . R – C/DVD Device (MMC-6)						
. . O – Optical Memory Block Device (SBC)						
. . . M – Media Changer Device (SMC-3)						
. . . A – Storage Array Device (SCC-2)						
. . . E – SCSI Enclosure Services device (SES)						
. . . B – Simplified Direct-Access (Reduced Block) device (RBC)						
. . . K – Optical Card Reader/Writer device (OCRW)						
. . . V – Automation/Device Interface device (ADC)						
. . . F – Object-based Storage Device (OSD)						
ASC	ASCQ	DTLPWROMAEBKVF	Description			
57h	00h	R	UNABLE TO RECOVER TABLE-OF-CONTENTS			
58h	00h	O	GENERATION DOES NOT EXIST			
59h	00h	O	UPDATED BLOCK READ			
5Ah	00h	DTLPWRO	BK	OPERATOR REQUEST OR STATE CHANGE INPUT		
5Ah	01h	DT	WROM	BK	OPERATOR MEDIUM REMOVAL REQUEST	
5Ah	02h	DT	WRO	A	BK	OPERATOR SELECTED WRITE PROTECT
5Ah	03h	DT	WRO	A	BK	OPERATOR SELECTED WRITE PERMIT
5Bh	00h	DTLPWROM	K	LOG EXCEPTION		
5Bh	01h	DTLPWROM	K	THRESHOLD CONDITION MET		
5Bh	02h	DTLPWROM	K	LOG COUNTER AT MAXIMUM		
5Bh	03h	DTLPWROM	K	LOG LIST CODES EXHAUSTED		
5Ch	00h	D	O	RPL STATUS CHANGE		
5Ch	01h	D	O	SPINDLES SYNCHRONIZED		
5Ch	02h	D	O	SPINDLES NOT SYNCHRONIZED		
5Dh	00h	DTLPWROMAEBKVF	FAILURE PREDICTION THRESHOLD EXCEEDED			
5Dh	01h	R	B	MEDIA FAILURE PREDICTION THRESHOLD EXCEEDED		
5Dh	02h	R	LOGICAL UNIT FAILURE PREDICTION THRESHOLD EXCEEDED			
5Dh	03h	R	SPARE AREA EXHAUSTION PREDICTION THRESHOLD EXCEEDED			
5Dh	10h	D	B	HARDWARE IMPENDING FAILURE GENERAL HARD DRIVE FAILURE		
5Dh	11h	D	B	HARDWARE IMPENDING FAILURE DRIVE ERROR RATE TOO HIGH		
5Dh	12h	D	B	HARDWARE IMPENDING FAILURE DATA ERROR RATE TOO HIGH		
5Dh	13h	D	B	HARDWARE IMPENDING FAILURE SEEK ERROR RATE TOO HIGH		
5Dh	14h	D	B	HARDWARE IMPENDING FAILURE TOO MANY BLOCK REASSIGNS		
5Dh	15h	D	B	HARDWARE IMPENDING FAILURE ACCESS TIMES TOO HIGH		
5Dh	16h	D	B	HARDWARE IMPENDING FAILURE START UNIT TIMES TOO HIGH		
5Dh	17h	D	B	HARDWARE IMPENDING FAILURE CHANNEL PARAMETRICS		
5Dh	18h	D	B	HARDWARE IMPENDING FAILURE CONTROLLER DETECTED		
5Dh	19h	D	B	HARDWARE IMPENDING FAILURE THROUGHPUT PERFORMANCE		
5Dh	1Ah	D	B	HARDWARE IMPENDING FAILURE SEEK TIME PERFORMANCE		
5Dh	1Bh	D	B	HARDWARE IMPENDING FAILURE SPIN-UP RETRY COUNT		
5Dh	1Ch	D	B	HARDWARE IMPENDING FAILURE DRIVE CALIBRATION RETRY COUNT		
5Dh	20h	D	B	CONTROLLER IMPENDING FAILURE GENERAL HARD DRIVE FAILURE		
5Dh	21h	D	B	CONTROLLER IMPENDING FAILURE DRIVE ERROR RATE TOO HIGH		
5Dh	22h	D	B	CONTROLLER IMPENDING FAILURE DATA ERROR RATE TOO HIGH		
5Dh	23h	D	B	CONTROLLER IMPENDING FAILURE SEEK ERROR RATE TOO HIGH		
5Dh	24h	D	B	CONTROLLER IMPENDING FAILURE TOO MANY BLOCK REASSIGNS		
5Dh	25h	D	B	CONTROLLER IMPENDING FAILURE ACCESS TIMES TOO HIGH		
5Dh	26h	D	B	CONTROLLER IMPENDING FAILURE START UNIT TIMES TOO HIGH		
5Dh	27h	D	B	CONTROLLER IMPENDING FAILURE CHANNEL PARAMETRICS		
5Dh	28h	D	B	CONTROLLER IMPENDING FAILURE CONTROLLER DETECTED		
5Dh	29h	D	B	CONTROLLER IMPENDING FAILURE THROUGHPUT PERFORMANCE		

**Table D.1 — ASC and ASCQ assignments** (part 13 of 17)

D – Direct Access Block Device (SBC-3)					Device Column key
. T – Sequential Access Device (SSC-3)					blank = code not used
. L – Printer Device (SSC)					not blank = code used
. P – Processor Device (SPC-2)					
. . W – Write Once Block Device (SBC)					
. . R – C/DVD Device (MMC-6)					
. . O – Optical Memory Block Device (SBC)					
. . M – Media Changer Device (SMC-3)					
. . A – Storage Array Device (SCC-2)					
. . E – SCSI Enclosure Services device (SES)					
. . B – Simplified Direct-Access (Reduced Block) device (RBC)					
. . K – Optical Card Reader/Writer device (OCRW)					
. . V – Automation/Device Interface device (ADC)					
. . F – Object-based Storage Device (OSD)					
ASC	ASCQ	DTLPWROMAEBKVF	Description		
5Dh	2Ah	D	B	CONTROLLER IMPENDING FAILURE SEEK TIME PERFORMANCE	
5Dh	2Bh	D	B	CONTROLLER IMPENDING FAILURE SPIN-UP RETRY COUNT	
5Dh	2Ch	D	B	CONTROLLER IMPENDING FAILURE DRIVE CALIBRATION RETRY COUNT	
5Dh	30h	D	B	DATA CHANNEL IMPENDING FAILURE GENERAL HARD DRIVE FAILURE	
5Dh	31h	D	B	DATA CHANNEL IMPENDING FAILURE DRIVE ERROR RATE TOO HIGH	
5Dh	32h	D	B	DATA CHANNEL IMPENDING FAILURE DATA ERROR RATE TOO HIGH	
5Dh	33h	D	B	DATA CHANNEL IMPENDING FAILURE SEEK ERROR RATE TOO HIGH	
5Dh	34h	D	B	DATA CHANNEL IMPENDING FAILURE TOO MANY BLOCK REASSIGNS	
5Dh	35h	D	B	DATA CHANNEL IMPENDING FAILURE ACCESS TIMES TOO HIGH	
5Dh	36h	D	B	DATA CHANNEL IMPENDING FAILURE START UNIT TIMES TOO HIGH	
5Dh	37h	D	B	DATA CHANNEL IMPENDING FAILURE CHANNEL PARAMETRICS	
5Dh	38h	D	B	DATA CHANNEL IMPENDING FAILURE CONTROLLER DETECTED	
5Dh	39h	D	B	DATA CHANNEL IMPENDING FAILURE THROUGHPUT PERFORMANCE	
5Dh	3Ah	D	B	DATA CHANNEL IMPENDING FAILURE SEEK TIME PERFORMANCE	
5Dh	3Bh	D	B	DATA CHANNEL IMPENDING FAILURE SPIN-UP RETRY COUNT	
5Dh	3Ch	D	B	DATA CHANNEL IMPENDING FAILURE DRIVE CALIBRATION RETRY COUNT	
5Dh	40h	D	B	SERVO IMPENDING FAILURE GENERAL HARD DRIVE FAILURE	
5Dh	41h	D	B	SERVO IMPENDING FAILURE DRIVE ERROR RATE TOO HIGH	
5Dh	42h	D	B	SERVO IMPENDING FAILURE DATA ERROR RATE TOO HIGH	
5Dh	43h	D	B	SERVO IMPENDING FAILURE SEEK ERROR RATE TOO HIGH	
5Dh	44h	D	B	SERVO IMPENDING FAILURE TOO MANY BLOCK REASSIGNS	
5Dh	45h	D	B	SERVO IMPENDING FAILURE ACCESS TIMES TOO HIGH	
5Dh	46h	D	B	SERVO IMPENDING FAILURE START UNIT TIMES TOO HIGH	
5Dh	47h	D	B	SERVO IMPENDING FAILURE CHANNEL PARAMETRICS	
5Dh	48h	D	B	SERVO IMPENDING FAILURE CONTROLLER DETECTED	
5Dh	49h	D	B	SERVO IMPENDING FAILURE THROUGHPUT PERFORMANCE	
5Dh	4Ah	D	B	SERVO IMPENDING FAILURE SEEK TIME PERFORMANCE	
5Dh	4Bh	D	B	SERVO IMPENDING FAILURE SPIN-UP RETRY COUNT	
5Dh	4Ch	D	B	SERVO IMPENDING FAILURE DRIVE CALIBRATION RETRY COUNT	
5Dh	50h	D	B	SPINDLE IMPENDING FAILURE GENERAL HARD DRIVE FAILURE	
5Dh	51h	D	B	SPINDLE IMPENDING FAILURE DRIVE ERROR RATE TOO HIGH	
5Dh	52h	D	B	SPINDLE IMPENDING FAILURE DATA ERROR RATE TOO HIGH	
5Dh	53h	D	B	SPINDLE IMPENDING FAILURE SEEK ERROR RATE TOO HIGH	
5Dh	54h	D	B	SPINDLE IMPENDING FAILURE TOO MANY BLOCK REASSIGNS	
5Dh	55h	D	B	SPINDLE IMPENDING FAILURE ACCESS TIMES TOO HIGH	
5Dh	56h	D	B	SPINDLE IMPENDING FAILURE START UNIT TIMES TOO HIGH	
5Dh	57h	D	B	SPINDLE IMPENDING FAILURE CHANNEL PARAMETRICS	
5Dh	58h	D	B	SPINDLE IMPENDING FAILURE CONTROLLER DETECTED	
5Dh	59h	D	B	SPINDLE IMPENDING FAILURE THROUGHPUT PERFORMANCE	
5Dh	5Ah	D	B	SPINDLE IMPENDING FAILURE SEEK TIME PERFORMANCE	
5Dh	5Bh	D	B	SPINDLE IMPENDING FAILURE SPIN-UP RETRY COUNT	



Table D.1 — ASC and ASCQ assignments (part 14 of 17)

		D – Direct Access Block Device (SBC-3) . T – Sequential Access Device (SSC-3) . L – Printer Device (SSC) . P – Processor Device (SPC-2) . . W – Write Once Block Device (SBC) . . R – C/DVD Device (MMC-6) . . O – Optical Memory Block Device (SBC) . . . M – Media Changer Device (SMC-3) . . . A – Storage Array Device (SCC-2) . . . E – SCSI Enclosure Services device (SES) . . . . B – Simplified Direct-Access (Reduced Block) device (RBC) . . . . K – Optical Card Reader/Writer device (OCRW) . . . . V – Automation/Device Interface device (ADC) . . . . F – Object-based Storage Device (OSD)					<u>Device Column key</u> blank = code not used not blank = code used
ASC	ASCQ	DTLPWROMAEBKVF	Description				
5Dh	5Ch	D	B	SPINDLE IMPENDING FAILURE DRIVE CALIBRATION RETRY COUNT			
5Dh	60h	D	B	FIRMWARE IMPENDING FAILURE GENERAL HARD DRIVE FAILURE			
5Dh	61h	D	B	FIRMWARE IMPENDING FAILURE DRIVE ERROR RATE TOO HIGH			
5Dh	62h	D	B	FIRMWARE IMPENDING FAILURE DATA ERROR RATE TOO HIGH			
5Dh	63h	D	B	FIRMWARE IMPENDING FAILURE SEEK ERROR RATE TOO HIGH			
5Dh	64h	D	B	FIRMWARE IMPENDING FAILURE TOO MANY BLOCK REASSIGNS			
5Dh	65h	D	B	FIRMWARE IMPENDING FAILURE ACCESS TIMES TOO HIGH			
5Dh	66h	D	B	FIRMWARE IMPENDING FAILURE START UNIT TIMES TOO HIGH			
5Dh	67h	D	B	FIRMWARE IMPENDING FAILURE CHANNEL PARAMETERS			
5Dh	68h	D	B	FIRMWARE IMPENDING FAILURE CONTROLLER DETECTED			
5Dh	69h	D	B	FIRMWARE IMPENDING FAILURE THROUGHPUT PERFORMANCE			
5Dh	6Ah	D	B	FIRMWARE IMPENDING FAILURE SEEK TIME PERFORMANCE			
5Dh	6Bh	D	B	FIRMWARE IMPENDING FAILURE SPIN-UP RETRY COUNT			
5Dh	6Ch	D	B	FIRMWARE IMPENDING FAILURE DRIVE CALIBRATION RETRY COUNT			
5Dh	FFh	DTLPWROMAEBKVF		FAILURE PREDICTION THRESHOLD EXCEEDED (FALSE)			
5Eh	00h	DTLPWRO	A	K	LOW POWER CONDITION ON		
5Eh	01h	DTLPWRO	A	K	IDLE CONDITION ACTIVATED BY TIMER		
5Eh	02h	DTLPWRO	A	K	STANDBY CONDITION ACTIVATED BY TIMER		
5Eh	03h	DTLPWRO	A	K	IDLE CONDITION ACTIVATED BY COMMAND		
5Eh	04h	DTLPWRO	A	K	STANDBY CONDITION ACTIVATED BY COMMAND		
5Eh	41h			B	POWER STATE CHANGE TO ACTIVE		
5Eh	42h			B	POWER STATE CHANGE TO IDLE		
5Eh	43h			B	POWER STATE CHANGE TO STANDBY		
5Eh	45h			B	POWER STATE CHANGE TO SLEEP		
5Eh	47h			BK	POWER STATE CHANGE TO DEVICE CONTROL		
5Fh	00h						
60h	00h				LAMP FAILURE		
61h	00h				VIDEO ACQUISITION ERROR		
61h	01h				UNABLE TO ACQUIRE VIDEO		
61h	02h				OUT OF FOCUS		
62h	00h				SCAN HEAD POSITIONING ERROR		
63h	00h		R		END OF USER AREA ENCOUNTERED ON THIS TRACK		
63h	01h		R		PACKET DOES NOT FIT IN AVAILABLE SPACE		
64h	00h		R		ILLEGAL MODE FOR THIS TRACK		
64h	01h		R		INVALID PACKET SIZE		
65h	00h	DTLPWROMAEBKVF			VOLTAGE FAULT		
66h	00h				AUTOMATIC DOCUMENT FEEDER COVER UP		
66h	01h				AUTOMATIC DOCUMENT FEEDER LIFT UP		
66h	02h				DOCUMENT JAM IN AUTOMATIC DOCUMENT FEEDER		
66h	03h				DOCUMENT MISS FEED AUTOMATIC IN DOCUMENT FEEDER		
67h	00h		A		CONFIGURATION FAILURE		

**Table D.1 — ASC and ASCQ assignments** (part 15 of 17)

D – Direct Access Block Device (SBC-3)														Device Column key	
. T – Sequential Access Device (SSC-3)														blank = code not used	
. L – Printer Device (SSC)														not blank = code used	
. P – Processor Device (SPC-2)															
. W – Write Once Block Device (SBC)															
. R – C/DVD Device (MMC-6)															
. O – Optical Memory Block Device (SBC)															
. M – Media Changer Device (SMC-3)															
. A – Storage Array Device (SCC-2)															
. E – SCSI Enclosure Services device (SES)															
. B – Simplified Direct-Access (Reduced Block) device (RBC)															
. K – Optical Card Reader/Writer device (OCRW)															
. V – Automation/Device Interface device (ADC)															
. F – Object-based Storage Device (OSD)															
ASC	ASCQ	DT	L	P	W	R	O	M	A	E	B	K	V	F	Description
67h	01h									A					CONFIGURATION OF INCAPABLE LOGICAL UNITS FAILED
67h	02h									A					ADD LOGICAL UNIT FAILED
67h	03h									A					MODIFICATION OF LOGICAL UNIT FAILED
67h	04h									A					EXCHANGE OF LOGICAL UNIT FAILED
67h	05h									A					REMOVE OF LOGICAL UNIT FAILED
67h	06h									A					ATTACHMENT OF LOGICAL UNIT FAILED
67h	07h									A					CREATION OF LOGICAL UNIT FAILED
67h	08h									A					ASSIGN FAILURE OCCURRED
67h	09h									A					MULTIPLY ASSIGNED LOGICAL UNIT
67h	0Ah	DT	L	P	W	R	O	M	A	E	B	K	V	F	SET TARGET PORT GROUPS COMMAND FAILED
67h	0Bh	DT									B				ATA DEVICE FEATURE NOT ENABLED
68h	00h									A					LOGICAL UNIT NOT CONFIGURED
69h	00h									A					DATA LOSS ON LOGICAL UNIT
69h	01h									A					MULTIPLE LOGICAL UNIT FAILURES
69h	02h									A					PARITY/DATA MISMATCH
6Ah	00h									A					INFORMATIONAL, REFER TO LOG
6Bh	00h									A					STATE CHANGE HAS OCCURRED
6Bh	01h									A					REDUNDANCY LEVEL GOT BETTER
6Bh	02h									A					REDUNDANCY LEVEL GOT WORSE
6Ch	00h									A					REBUILD FAILURE OCCURRED
6Dh	00h									A					RECALCULATE FAILURE OCCURRED
6Eh	00h									A					COMMAND TO LOGICAL UNIT FAILED
6Fh	00h					R									COPY PROTECTION KEY EXCHANGE FAILURE - AUTHENTICATION FAILURE
6Fh	01h					R									COPY PROTECTION KEY EXCHANGE FAILURE - KEY NOT PRESENT
6Fh	02h					R									COPY PROTECTION KEY EXCHANGE FAILURE - KEY NOT ESTABLISHED
6Fh	03h					R									READ OF SCRAMBLED SECTOR WITHOUT AUTHENTICATION
6Fh	04h					R									MEDIA REGION CODE IS MISMATCHED TO LOGICAL UNIT REGION
6Fh	05h					R									DRIVE REGION MUST BE PERMANENT/REGION RESET COUNT ERROR
6Fh	06h					R									INSUFFICIENT BLOCK COUNT FOR BINDING NONCE RECORDING
6Fh	07h					R									CONFLICT IN BINDING NONCE RECORDING
70h	NNh	T													DECOMPRESSION EXCEPTION SHORT ALGORITHM ID OF NN
71h	00h	T													DECOMPRESSION EXCEPTION LONG ALGORITHM ID
72h	00h					R									SESSION FIXATION ERROR
72h	01h					R									SESSION FIXATION ERROR WRITING LEAD-IN
72h	02h					R									SESSION FIXATION ERROR WRITING LEAD-OUT
72h	03h					R									SESSION FIXATION ERROR - INCOMPLETE TRACK IN SESSION
72h	04h					R									EMPTY OR PARTIALLY WRITTEN RESERVED TRACK
72h	05h					R									NO MORE TRACK RESERVATIONS ALLOWED
72h	06h					R									RMZ EXTENSION IS NOT ALLOWED
72h	07h					R									NO MORE TEST ZONE EXTENSIONS ARE ALLOWED
73h	00h					R									CD CONTROL ERROR

Table D.1 — ASC and ASCQ assignments (part 16 of 17)

D – Direct Access Block Device (SBC-3)														Device Column key	
. T – Sequential Access Device (SSC-3)														blank = code not used	
. L – Printer Device (SSC)														not blank = code used	
. P – Processor Device (SPC-2)															
. . W – Write Once Block Device (SBC)															
. . R – C/DVD Device (MMC-6)															
. . O – Optical Memory Block Device (SBC)															
. . . M – Media Changer Device (SMC-3)															
. . . A – Storage Array Device (SCC-2)															
. . . E – SCSI Enclosure Services device (SES)															
. . . B – Simplified Direct-Access (Reduced Block) device (RBC)															
. . . K – Optical Card Reader/Writer device (OCRW)															
. . . V – Automation/Device Interface device (ADC)															
. . . F – Object-based Storage Device (OSD)															
. . . . .															
ASC	ASCQ	DT	L	P	W	R	O	M	A	E	B	K	V	F	Description
73h	01h					R									POWER CALIBRATION AREA ALMOST FULL
73h	02h					R									POWER CALIBRATION AREA IS FULL
73h	03h					R									POWER CALIBRATION AREA ERROR
73h	04h					R									PROGRAM MEMORY AREA UPDATE FAILURE
73h	05h					R									PROGRAM MEMORY AREA IS FULL
73h	06h					R									RMA/PMA IS ALMOST FULL
73h	10h					R									CURRENT POWER CALIBRATION AREA ALMOST FULL
73h	11h					R									CURRENT POWER CALIBRATION AREA IS FULL
73h	17h					R									RDZ IS FULL
74h	00h		T												SECURITY ERROR
74h	01h		T												UNABLE TO DECRYPT DATA
74h	02h		T												UNENCRYPTED DATA ENCOUNTERED WHILE DECRYPTING
74h	03h		T												INCORRECT DATA ENCRYPTION KEY
74h	04h		T												CRYPTOGRAPHIC INTEGRITY VALIDATION FAILED
74h	05h		T												ERROR DECRYPTING DATA
74h	06h		T												UNKNOWN SIGNATURE VERIFICATION KEY
74h	07h		T												ENCRYPTION PARAMETERS NOT USEABLE
74h	08h	DT				R		M		E			V	F	DIGITAL SIGNATURE VALIDATION FAILURE
74h	09h		T												ENCRYPTION MODE MISMATCH ON READ
74h	0Ah		T												ENCRYPTED BLOCK NOT RAW READ ENABLED
74h	0Bh		T												INCORRECT ENCRYPTION PARAMETERS
74h	0Ch	DT				R			M	A	E	B	K	V	UNABLE TO DECRYPT PARAMETER LIST
74h	0Dh		T												ENCRYPTION ALGORITHM DISABLED
74h	10h	DT				R			M	A	E	B	K	V	SA CREATION PARAMETER VALUE INVALID
74h	11h	DT				R			M	A	E	B	K	V	SA CREATION PARAMETER VALUE REJECTED
74h	12h	DT				R			M	A	E	B	K	V	INVALID SA USAGE
74h	21h		T												DATA ENCRYPTION CONFIGURATION PREVENTED
74h	30h	DT				R			M	A	E	B	K	V	SA CREATION PARAMETER NOT SUPPORTED
74h	40h	DT				R			M	A	E	B	K	V	AUTHENTICATION FAILED
74h	61h										V				EXTERNAL DATA ENCRYPTION KEY MANAGER ACCESS ERROR
74h	62h										V				EXTERNAL DATA ENCRYPTION KEY MANAGER ERROR
74h	63h										V				EXTERNAL DATA ENCRYPTION KEY NOT FOUND
74h	64h										V				EXTERNAL DATA ENCRYPTION REQUEST NOT AUTHORIZED
74h	6Eh		T												EXTERNAL DATA ENCRYPTION CONTROL TIMEOUT
74h	6Fh		T												EXTERNAL DATA ENCRYPTION CONTROL ERROR
74h	71h	DT				R		M		E			V		LOGICAL UNIT ACCESS NOT AUTHORIZED
74h	79h		D												SECURITY CONFLICT IN TRANSLATED DEVICE
75h	00h														
76h	00h														
77h	00h														
78h	00h														

**Table D.1 — ASC and ASCQ assignments** (part 17 of 17)

D – Direct Access Block Device (SBC-3)														<u>Device Column key</u> blank = code not used not blank = code used	
. T – Sequential Access Device (SSC-3)															
. L – Printer Device (SSC)															
. P – Processor Device (SPC-2)															
. . W – Write Once Block Device (SBC)															
. . R – C/DVD Device (MMC-6)															
. . O – Optical Memory Block Device (SBC)															
. . . M – Media Changer Device (SMC-3)															
. . . A – Storage Array Device (SCC-2)															
. . . E – SCSI Enclosure Services device (SES)															
. . . B – Simplified Direct-Access (Reduced Block) device (RBC)															
. . . K – Optical Card Reader/Writer device (OCRW)															
. . . V – Automation/Device Interface device (ADC)															
. . . F – Object-based Storage Device (OSD)															
. . . . .															
ASC	ASCQ	DT	L	P	W	R	O	M	A	E	B	K	V	F	Description
79h	00h														
7Ah	00h														
7Bh	00h														
7Ch	00h														
7Dh	00h														
7Eh	00h														
7Fh	00h														
80h	xxh													\	
Through														>	Vendor specific
FFh	xxh													/	
xxh	80h													\	
Through														>	Vendor specific qualification of standard ASC
xxh	FFh													/	
All codes not shown are reserved.															

## D.3 Operation codes

### D.3.1 Operation codes

Table D.2 is a numerical order listing of the command operation codes.

**Table D.2 — Operation codes** (part 1 of 6)

D – Direct Access Block Device (SBC-3)														Device Column key	
. T – Sequential Access Device (SSC-3)														M = Mandatory	
. L – Printer Device (SSC)														O = Optional	
. P – Processor Device (SPC-2)														V = Vendor specific	
. . W – Write Once Block Device (SBC)														Z = Obsolete	
. . R – C/DVD Device (MMC-6)															
. . . O – Optical Memory Block Device (SBC)															
. . . M – Media Changer Device (SMC-3)															
. . . A – Storage Array Device (SCC-2)															
. . . E – SCSI Enclosure Services device (SES)															
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)															
. . . . K – Optical Card Reader/Writer device (OCRW)															
. . . . V – Automation/Device Interface device (ADC)															
. . . . F – Object-based Storage Device (OSD)															
OP	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
00h	M	M	M	M	M	M	M	M	M	M	M	M	M	M	TEST UNIT READY
01h		M													REWIND
01h	Z		V		Z	Z	Z	Z							REZERO UNIT
02h		V	V	V	V	V	V	V							
03h	M	M	M	M	M	M	M	M	M	O	M	M	M		REQUEST SENSE
04h	M							O	O						FORMAT UNIT
04h		O													FORMAT MEDIUM
04h			O												FORMAT
05h		V	M	V	V	V	V	V							READ BLOCK LIMITS
06h		V	V	V	V	V	V	V							
07h		O	V			O			O	V					REASSIGN BLOCKS
07h									O						INITIALIZE ELEMENT STATUS
08h		M	O	V											READ(6)
08h			O												RECEIVE
08h															GET MESSAGE(6)
09h		V	V	V	V	V	V	V							
0Ah		O	O			O			O	V					WRITE(6)
0Ah				M											SEND(6)
0Ah															SEND MESSAGE(6)
0Ah				M											PRINT
0Bh	Z					Z	O	Z	V						SEEK(6)
0Bh			O												SET CAPACITY
0Bh				O											SLEW AND PRINT
0Ch		V	V	V	V	V	V	V							
0Dh		V	V	V	V	V	V	V							
0Eh		V	V	V	V	V	V	V							
0Fh		V	O	V	V	V	V	V							READ REVERSE(6)
10h		V	M			V	V	V							WRITE FILEMARKS(6)
10h			O												SYNCHRONIZE BUFFER
11h		V	M	V	V	V	V								SPACE(6)
12h	M	M	M	M	M	M	M	M	M	M	M	M	M		INQUIRY
13h		V			V	V	V	V							
13h			O												VERIFY(6)
14h		V	O	O	V	V	V								RECOVER BUFFERED DATA
15h	O	M	O		O	O	O	O		O	O				MODE SELECT(6)

Table D.2 — Operation codes (part 2 of 6)

D – Direct Access Block Device (SBC-3) . T – Sequential Access Device (SSC-3) . L – Printer Device (SSC) . P – Processor Device (SPC-2) . W – Write Once Block Device (SBC) . R – C/DVD Device (MMC-6) . O – Optical Memory Block Device (SBC) . M – Media Changer Device (SMC-3) . A – Storage Array Device (SCC-2) . E – SCSI Enclosure Services device (SES) . B – Simplified Direct-Access (Reduced Block) device (RBC) . K – Optical Card Reader/Writer device (OCRW) . V – Automation/Device Interface device (ADC) . F – Object-based Storage Device (OSD)													<u>Device Column key</u> M = Mandatory O = Optional V = Vendor specific Z = Obsolete	
OP	D	T	L	P	W	R	O	A	E	B	K	V	F	Description
16h	Z	Z	M	Z	O					O				RESERVE(6)
16h									Z					RESERVE ELEMENT(6)
17h	Z	Z	M	Z	O					O				RELEASE(6)
17h									Z					RELEASE ELEMENT(6)
18h	Z	Z	Z	Z	Z	O	Z	O				Z		COPY
19h	V	M	V	V	V	V								ERASE(6)
1Ah	O	M	O		O	O	O	O	O	O				MODE SENSE(6)
1Bh	O			O	O	O	O		M	O	O			START STOP UNIT
1Bh	O										M			LOAD UNLOAD
1Bh														SCAN
1Bh	O													STOP PRINT
1Bh							O							OPEN/CLOSE IMPORT/EXPORT ELEMENT
1Ch	O	O	O	O	O	O	O	O	O	O	O	O	O	RECEIVE DIAGNOSTIC RESULTS
1Dh	M	M	M	M	M	M	M	M	M	M	M	M	M	SEND DIAGNOSTIC
1Eh	O	O		O	O	O	O			O	O			PREVENT ALLOW MEDIUM REMOVAL
1Fh														
20h	V				V	V	V				V			
21h	V				V	V	V				V			
22h	V				V	V	V				V			
23h	V				V	V					V			
23h					O									READ FORMAT CAPACITIES
24h	V				V	V								SET WINDOW
25h	M				M	M					M			READ CAPACITY(10)
25h					O									READ CAPACITY
25h											M			READ CARD CAPACITY
25h														GET WINDOW
26h	V				V	V								
27h	V				V	V								
28h	M				M	O	M				M	M		READ(10)
28h														GET MESSAGE(10)
29h	V				V	V	O							READ GENERATION
2Ah	O				M	O	M				M	O		WRITE(10)
2Ah														SEND(10)
2Ah														SEND MESSAGE(10)
2Bh	Z				O	O	O				O			SEEK(10)
2Bh	O													LOCATE(10)
2Bh							O							POSITION TO ELEMENT
2Ch	V				O	O								ERASE(10)
2Dh					O									READ UPDATED BLOCK
2Dh	V													
2Eh	O				O	O	O				M	O		WRITE AND VERIFY(10)
2Fh	O				O	O	O							VERIFY(10)

Table D.2 — Operation codes (part 3 of 6)

D – Direct Access Block Device (SBC-3) . T – Sequential Access Device (SSC-3) . L – Printer Device (SSC) . P – Processor Device (SPC-2) . . W – Write Once Block Device (SBC) . . R – C/DVD Device (MMC-6) . . . O – Optical Memory Block Device (SBC) . . . M – Media Changer Device (SMC-3) . . . A – Storage Array Device (SCC-2) . . . E – SCSI Enclosure Services device (SES) . . . . B – Simplified Direct-Access (Reduced Block) device (RBC) . . . . K – Optical Card Reader/Writer device (OCRW) . . . . V – Automation/Device Interface device (ADC) . . . . F – Object-based Storage Device (OSD)														<u>Device Column key</u> M = Mandatory O = Optional V = Vendor specific Z = Obsolete	
OP	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
30h	Z				Z	Z	Z								SEARCH DATA HIGH(10)
31h	Z				Z	Z	Z								SEARCH DATA EQUAL(10)
31h															OBJECT POSITION
32h	Z				Z	Z	Z								SEARCH DATA LOW(10)
33h	Z				O	Z	O								SET LIMITS(10)
34h	O				O	O						O			PRE-FETCH(10)
34h		M													READ POSITION
34h															GET DATA BUFFER STATUS
35h	O				O	O	O					M	O		SYNCHRONIZE CACHE(10)
36h	Z				O	O							O		LOCK UNLOCK CACHE(10)
37h	O					O									READ DEFECT DATA(10)
37h							O								INITIALIZE ELEMENT STATUS WITH RANGE
38h					O	O							O		MEDIUM SCAN
39h	Z	Z	Z	Z	O	Z	O					Z			COMPARE
3Ah	Z	Z	Z	Z	O	Z	O					Z			COPY AND VERIFY
3Bh	O	O	O	O	O	O	O	O	O	O	O	O	O	O	WRITE BUFFER
3Ch	O	O	O	O	O	O	O	O	O	O	O	O	O	O	READ BUFFER
3Dh						O									UPDATE BLOCK
3Eh	O				O	O									READ LONG(10)
3Fh	O				O	O									WRITE LONG(10)
40h	Z	Z	Z	Z	O	Z	O	Z							CHANGE DEFINITION
41h	O														WRITE SAME(10)
42h						O									READ SUB-CHANNEL
43h						O									READ TOC/PMA/ATIP
44h		M											M		REPORT DENSITY SUPPORT
44h															READ HEADER
45h						O									PLAY AUDIO(10)
46h						M									GET CONFIGURATION
47h						O									PLAY AUDIO MSF
48h															
49h															
4Ah						M									GET EVENT STATUS NOTIFICATION
4Bh						O									PAUSE/RESUME
4Ch	O	O	O	O	O	O	O	O	O	O	O	O	O	O	LOG SELECT
4Dh	O	O	O	O	O	O	O	O	O	O	O	O	O	M	LOG SENSE
4Eh						O									STOP PLAY/SCAN
4Fh															
50h	O														XDWRITE(10)
51h	O														XPWRITE(10)
51h						O									READ DISC INFORMATION
52h	O														XDREAD(10)
52h						O									READ TRACK INFORMATION

Table D.2 — Operation codes (part 4 of 6)

D – Direct Access Block Device (SBC-3)														Device Column key	
. T – Sequential Access Device (SSC-3)														M = Mandatory	
. L – Printer Device (SSC)														O = Optional	
. P – Processor Device (SPC-2)														V = Vendor specific	
. W – Write Once Block Device (SBC)														Z = Obsolete	
. R – C/DVD Device (MMC-6)															
. O – Optical Memory Block Device (SBC)															
. M – Media Changer Device (SMC-3)															
. A – Storage Array Device (SCC-2)															
. E – SCSI Enclosure Services device (SES)															
. B – Simplified Direct-Access (Reduced Block) device (RBC)															
. K – Optical Card Reader/Writer device (OCRW)															
. V – Automation/Device Interface device (ADC)															
. F – Object-based Storage Device (OSD)															
OP	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
53h							O								RESERVE TRACK
54h							O								SEND OPC INFORMATION
55h	O	O	O	O	M	O	O	O	O	M	O	M	O	M	MODE SELECT(10)
56h	Z	Z	M	Z	O					O	O	O	Z		RESERVE(10)
56h										Z					RESERVE ELEMENT(10)
57h	Z	Z	M	Z	O					O	O	O	Z		RELEASE(10)
57h										Z					RELEASE ELEMENT(10)
58h							O								REPAIR TRACK
59h															
5Ah	O	O	O	O	M	O	O	O	O	M	O	M	O	M	MODE SENSE(10)
5Bh							O								CLOSE TRACK/SESSION
5Ch							O								READ BUFFER CAPACITY
5Dh							O								SEND CUE SHEET
5Eh	O	O	O	O	O	O				O	O	O		M	PERSISTENT RESERVE IN
5Fh	O	O	O	O	O	O				O	O	O		M	PERSISTENT RESERVE OUT
7Eh	O	O				O				O	O	O	O	O	extended CDB
7Fh	O													M	variable length CDB (more than 16 bytes)
80h	Z														XDWRITE EXTENDED(16)
80h	M														WRITE FILEMARKS(16)
81h	Z														REBUILD(16)
81h	O														READ REVERSE(16)
82h	Z														REGENERATE(16)
83h	O	O	O	O	O	O				O				O	EXTENDED COPY
84h	O	O	O	O	O	O				O				O	RECEIVE COPY RESULTS
85h	O					O								O	ATA COMMAND PASS THROUGH(16)
86h	O	O				O	O	O	O	O	O	O	O	O	ACCESS CONTROL IN
87h	O	O				O	O	O	O	O	O	O	O	O	ACCESS CONTROL OUT
88h	M	M				O	O							O	READ(16)
89h															
8Ah	O	M				O	O							O	WRITE(16)
8Bh	O														ORWRITE
8Ch	O	O				O	O			O	O			M	READ ATTRIBUTE
8Dh	O	O				O	O			O	O			O	WRITE ATTRIBUTE
8Eh	O					O	O							O	WRITE AND VERIFY(16)
8Fh	O	O				O	O							O	VERIFY(16)
90h	O					O	O							O	PRE-FETCH(16)
91h	O					O	O							O	SYNCHRONIZE CACHE(16)
91h	O														SPACE(16)
92h	Z					O	O								LOCK UNLOCK CACHE(16)
92h	O														LOCATE(16)
93h	O														WRITE SAME(16)
93h	M														ERASE(16)



Table D.2 — Operation codes (part 5 of 6)

D – Direct Access Block Device (SBC-3) . T – Sequential Access Device (SSC-3) . L – Printer Device (SSC) . P – Processor Device (SPC-2) . W – Write Once Block Device (SBC) . R – C/DVD Device (MMC-6) O – Optical Memory Block Device (SBC) M – Media Changer Device (SMC-3) A – Storage Array Device (SCC-2) E – SCSI Enclosure Services device (SES) B – Simplified Direct-Access (Reduced Block) device (RBC) K – Optical Card Reader/Writer device (OCRW) V – Automation/Device Interface device (ADC) F – Object-based Storage Device (OSD)														<u>Device Column key</u> M = Mandatory O = Optional V = Vendor specific Z = Obsolete	
OP	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
94h															[usage proposed by SCSI Socket Services project]
95h															[usage proposed by SCSI Socket Services project]
96h															[usage proposed by SCSI Socket Services project]
97h															[usage proposed by SCSI Socket Services project]
98h															
99h															
9Ah															
9Bh															
9Ch															
9Dh															
9Eh															SERVICE ACTION IN(16)
9Fh													M		SERVICE ACTION OUT(16)
A0h	M	M	O	O	O			O	M	M		O	M	O	REPORT LUNS
A1h								O							BLANK
A1h	O											O			ATA COMMAND PASS THROUGH(12)
A2h	O	O						O					O		SECURITY PROTOCOL IN
A3h	O	O	O		O		O	O	M	O	O	O	M		MAINTENANCE (IN)
A3h								O							SEND KEY
A4h	O	O	O		O		O	O	O	O	O	O	O	O	MAINTENANCE (OUT)
A4h								O							REPORT KEY
A5h		O			O		O	M							MOVE MEDIUM
A5h								O							PLAY AUDIO(12)
A6h										O					EXCHANGE MEDIUM
A6h								O							LOAD/UNLOAD C/DVD
A7h	Z	Z			O		O								MOVE MEDIUM ATTACHED
A7h								O							SET READ AHEAD
A8h	O				O	O	O								READ(12)
A8h															GET MESSAGE(12)
A9h													O		SERVICE ACTION OUT(12)
AAh	O				O	O	O								WRITE(12)
AAh															SEND MESSAGE(12)
ABh						O							O		SERVICE ACTION IN(12)
ACH								O							ERASE(12)
ACH								O							GET PERFORMANCE
ADh								O							READ DVD STRUCTURE
Aeh	O				O		O								WRITE AND VERIFY(12)
Afh	O				O	Z	O								VERIFY(12)
B0h					Z	Z	Z								SEARCH DATA HIGH(12)
B1h					Z	Z	Z								SEARCH DATA EQUAL(12)
B2h					Z	Z	Z								SEARCH DATA LOW(12)
B3h	Z				O	Z	O								SET LIMITS(12)
B4h	Z	Z			O	Z	O								READ ELEMENT STATUS ATTACHED

**Table D.2 — Operation codes (part 6 of 6)**

D – Direct Access Block Device (SBC-3)											<u>Device Column key</u>				
. T – Sequential Access Device (SSC-3)											M = Mandatory				
. L – Printer Device (SSC)											O = Optional				
. P – Processor Device (SPC-2)											V = Vendor specific				
. W – Write Once Block Device (SBC)											Z = Obsolete				
. R – C/DVD Device (MMC-6)															
. O – Optical Memory Block Device (SBC)															
. M – Media Changer Device (SMC-3)															
. A – Storage Array Device (SCC-2)															
. E – SCSI Enclosure Services device (SES)															
. B – Simplified Direct-Access (Reduced Block) device (RBC)															
. K – Optical Card Reader/Writer device (OCRW)															
. V – Automation/Device Interface device (ADC)															
. F – Object-based Storage Device (OSD)															
OP	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
B5h	O	O				O								O	SECURITY PROTOCOL OUT
B5h							O								REQUEST VOLUME ELEMENT ADDRESS
B6h							O								SEND VOLUME TAG
B6h							O								SET STREAMING
B7h	O						O								READ DEFECT DATA(12)
B8h		O			O	Z	O	M							READ ELEMENT STATUS
B9h							O								READ CD MSF
BAh	O				O	O	O	M	O						REDUNDANCY GROUP (IN)
BAh							O								SCAN
BBh	O				O	O	O	O	O						REDUNDANCY GROUP (OUT)
BBh							O								SET CD SPEED
BCh	O				O	O	O	M	O						SPARE (IN)
BDh	O				O	O	O	O	O						SPARE (OUT)
BDh							O								MECHANISM STATUS
BEh	O				O	O	O	M	O						VOLUME SET (IN)
BEh							O								READ CD
BFh	O				O	O	O	O	O						VOLUME SET (OUT)
BFh							O								SEND DVD STRUCTURE

Device Column key

M = Mandatory

O = Optional

V = Vendor specific

Z = Obsolete

### D.3.2 Additional operation codes for devices with the EncServ bit set to one

Table D.3 is a numerical order listing of the additional command operation codes used by devices that have the EncServ bit set to one in their standard INQUIRY data. The operation codes listed in table D.3 are in addition to the operation codes listed in D.3.1 for the device type indicated by the standard INQUIRY data having the EncServ bit set to one.

**Table D.3 — Additional operation codes for devices with the EncServ bit set to one**

D – Direct Access Block Device (SBC-3) . T – Sequential Access Device (SSC-3) . L – Printer Device (SSC) . P – Processor Device (SPC-2) . W – Write Once Block Device (SBC) . R – C/DVD Device (MMC-6) . O – Optical Memory Block Device (SBC) . M – Media Changer Device (SMC-3) . A – Storage Array Device (SCC-2) . E – SCSI Enclosure Services device (SES) . B – Simplified Direct-Access (Reduced Block) device (RBC) . K – Optical Card Reader/Writer device (OCRW) . V – Automation/Device Interface device (ADC) . F – Object-based Storage Device (OSD)			<u>Device Column key</u> M = Mandatory O = Optional V = Vendor specific Z = Obsolete	
OP	D T L P W R O M A E B K V F	Description		
1C	MMMMMMMMMMMMMMMM	RECEIVE DIAGNOSTIC RESULTS		
1D	MMMMMMMMMMMMMMMM	SEND DIAGNOSTIC		

### D.3.3 MAINTENANCE (IN) and MAINTENANCE (OUT) service actions

The assignment of service action codes for the MAINTENANCE (IN) and MAINTENANCE (OUT) operation codes by this standard is shown in table D.4. The MAINTENANCE (IN) and MAINTENANCE (OUT) service actions that may be assigned by other command standards are noted as restricted but their specific usage is not described.

**Table D.4 — MAINTENANCE (IN) and MAINTENANCE (OUT) service actions**

Service Action	Description
MAINTENANCE (IN) [operation code A3h]	
00h to 04h	Restricted
05h	REPORT IDENTIFYING INFORMATION
06h to 09h	Restricted
0Ah	REPORT TARGET PORT GROUPS
0Bh	REPORT ALIASES
0Ch	REPORT SUPPORTED OPERATION CODES
0Dh	REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS
0Eh	REPORT PRIORITY
0Fh	REPORT TIMESTAMP
10h	MANAGEMENT PROTOCOL IN
11h to 1Eh	Reserved
1Fh	Vendor specific
MAINTENANCE (OUT) [operation code A4h]	
00h to 05h	Restricted
06h	SET IDENTIFYING INFORMATION
07h to 09h	Restricted
0Ah	SET TARGET PORT GROUPS
0Bh	CHANGE ALIASES
0Ch to 0Dh	Reserved
0Eh	SET PRIORITY
0Fh	SET TIMESTAMP
10h	MANAGEMENT PROTOCOL OUT
11h to 1Eh	Reserved
1Fh	Vendor specific

### D.3.4 SERVICE ACTION IN and SERVICE ACTION OUT service actions

The assignment of service action codes for the SERVICE ACTION IN(12) and SERVICE ACTION OUT(12) operation codes by this standard is shown in table D.5.

**Table D.5 — SERVICE ACTION IN(12) and SERVICE ACTION OUT(12) service actions**

Service Action	Description
SERVICE ACTION IN(12) [operation code ABh]	
00h	Reserved
01h	READ MEDIA SERIAL NUMBER
02h to 1Fh	Reserved
SERVICE ACTION OUT(12) [operation code A9h]	
00h to 1Eh	Reserved
1Fh	Restricted

The assignment of service action codes for the SERVICE ACTION IN(16) and SERVICE ACTION OUT(16) operation codes by this standard is shown in table D.6. The SERVICE ACTION IN(16) and SERVICE ACTION OUT(16) service actions that may be assigned by other command standards are noted as restricted but their specific usage is not described.

**Table D.6 — SERVICE ACTION IN(16) and SERVICE ACTION OUT(16) service actions**

Service Action	Description
SERVICE ACTION IN(16) [operation code 9Eh]	
00h to 0Fh	Reserved
10h to 1Fh	Restricted
SERVICE ACTION OUT(16) [operation code 9Fh]	
00h to 0Fh	Reserved
10h to 1Fh	Restricted

### D.3.5 Variable length CDB service action codes

Only one operation code is assigned to the variable length CDB (see 4.3.3). Therefore, the service action code is effectively the operation code for variable length CDB uses. To allow command standards to assign uses of the variable length CDB without consulting this standard, ranges of service action codes are assigned to command sets as shown in table D.7.

**Table D.7 — Variable Length CDB Service Action Code Ranges**

Service Action Code Range	Doc.	Description
0000h to 07FFh	SBC-3	Direct access block device (e.g., magnetic disk)
0800h to 0FFFh	SSC-3	Sequential-access device (e.g., magnetic tape)
1000h to 17FFh	SSC	Printer device
1800h to 1FFFh	this standard	Commands for all device types (see table D.8)
2000h to 27FFh		Reserved
2800h to 2FFFh	MMC-5	CD-ROM device
3800h to 3FFFh		Reserved
4000h to 47FFh	SMC-3	Media changer device (e.g., jukeboxes)
5000h to 5FFFh		Defined by ASC IT8 (Graphic arts pre-press devices)
6000h to 67FFh	SCC-2	Storage array controller device (e.g., RAID)
7000h to 77FFh	RBC	Simplified direct-access device (e.g., magnetic disk)
7800h to 7FFFh	OCRW	Optical card reader/writer device
8800h to 8FFFh	OSD	Object-based Storage Device
3000h to 37FFh		Reserved
4800h to 4FFFh		Reserved
6800h to 6FFFh		Reserved
8000h to 87FFh		Reserved
9000h to F7FFh		Reserved
F800h to FFFFh		Vendor specific

The variable length CDB service action codes assigned by this standard are shown in table D.8.

**Table D.8 — Variable Length CDB Service Action Codes Used by All Device Types**

Service Action Code	Description
1800h	RECEIVE CREDENTIAL command
1801h to 1FFFh	Reserved

## D.4 Diagnostic page codes

Table D.9 is a numerical order listing of the diagnostic page codes.

**Table D.9 — Diagnostic page codes**

Diagnostic Page Code				Diagnostic Page Name
	DTLPWROMAE	BKVF		
	D – Direct Access Block Device (SBC-3) . T – Sequential Access Device (SSC-3) . L – Printer Device (SSC) . P – Processor Device (SPC-2) . . W – Write Once Block Device (SBC) . . R – C/DVD Device (MMC-6) . . O – Optical Memory Block Device (SBC) . . . M – Media Changer Device (SMC-3) . . . A – Storage Array Device (SCC-2) . . . E – SCSI Enclosure Services device (SES) . . . . B – Simplified Direct-Access (Reduced Block) device (RBC) . . . . K – Optical Card Reader/Writer device (OCRW) . . . . V – Automation/Device Interface device (ADC) . . . . F – Object-based Storage Device (OSD)			<u>Device Column key</u> blank = code not used not blank = code used
00h	D	T	L	Supported Diagnostic Pages
01h			E	Configuration
02h			E	Enclosure Status/Control
03h			E	Help Text
04h			E	String In/Out
05h			E	Threshold In/Out
06h			E	Obsolete
07h			E	Element Descriptor
08h			E	Short Enclosure Status
09h			E	Enclosure Busy
0Ah			E	Additional Element Status
0Bh			E	Subenclosure Help Text
0Ch			E	Subenclosure String In/Out
0Dh			E	Supported SES Diagnostic Pages
0Eh			E	Download Microcode Status/Control
0Fh			E	Subenclosure Nickname Status/Control
10h to 1Fh				SES vendor specific
3Fh				Protocol Specific diagnostic page
40h	D	W	O	Translate Address In/Out
41h	D	W	O	Device Status In/Out
80h to FFh				Vendor specific
<b>All codes not shown are restricted or reserved.</b>				

## D.5 Log page codes

Table D.10 is a numerical order listing of the log page codes.

**Table D.10 — Log page codes** (part 1 of 2)

Log Page Code	Log Subpage Code	D – Direct Access Block Device (SBC-3) . T – Sequential Access Device (SSC-3) . L – Printer Device (SSC) . P – Processor Device (SPC-2) . W – Write Once Block Device (SBC) . R – C/DVD Device (MMC-6) . O – Optical Memory Block Device (SBC) . M – Media Changer Device (SMC-3) . A – Storage Array Device (SCC-2) . E – SCSI Enclosure Services device (SES) . B – Simplified Direct-Access (Reduced Block) device (RBC) . K – Optical Card Reader/Writer device (OCRW) . V – Automation/Device Interface device (ADC) . F – Object-based Storage Device (OSD)		Log Page Name
		DTLPWROMAEBKVF		
00h	00h	DTLPWROMAE	KVF	Supported Log Pages
00h	FFh	DTLPWROMAE	KVF	Supported Log Pages and Subpages
01h to 3Fh	FFh	DTLPWROMAE	KVF	Supported Subpages
01h	00h	DTLPWRO A	KVF	Buffer Over-Run/Under-Run
02h	00h	DT WRO	KV	Write Error Counter
03h	00h	DT WRO	KV	Read Error Counter
04h	00h	T	V	Read Reverse Error Counter
05h	00h	DT WRO	KV	Verify Error Counter
06h	00h	DTLPWROMAE	KVF	Non-Medium Error
07h	00h	DTLPWROMAE	KVF	Last n Error Events
08h	00h	DT W O	V	Format Status
09h	00h to FFh	O		Reserved to the MS59 Std. (contact AIIM C21 comm.)
0Ah	00h to FFh	O		Reserved to the MS59 Std. (contact AIIM C21 comm.)
0Bh	00h	DTLPWROMAE	VF	Last n Deferred Error or Asynchronous Events
0Ch	00h	T	V	Sequential-Access Device
0Dh	00h	DTLPWROMAE	V	Temperature
0Eh	00h	DTLPWROMAE	V	Start-Stop Cycle Counter
0Fh	00h	DTLPWROMAE	V	Application Client
10h	00h	DTLPWROMAE	V	Self-Test Results
11h	00h	T	V	DT Device Status
12h	00h		V	TapeAlert Response
13h	00h		V	Requested Recovery
14h	00h	T	V	Device Statistics
15h	00h	D		Background Scan Results
15h	00h		V	Service Buffers Information
16h	00h	T		Tape Diagnostic Data
17h	00h	D		Non-Volatile Cache
18h	xxh	DTLPWROMAE	KV	Protocol Specific Port (see table D.11)
19h	00h	D		General Statistics and Performance
19h	01h	D		Group Statistics and Performance (1)
19h	02h	D		Group Statistics and Performance (2)
19h	03h	D		Group Statistics and Performance (3)
19h	04h	D		Group Statistics and Performance (4)
19h	05h	D		Group Statistics and Performance (5)
19h	06h	D		Group Statistics and Performance (6)
19h	07h	D		Group Statistics and Performance (7)



Table D.10 — Log page codes (part 2 of 2)

		D – Direct Access Block Device (SBC-3) . T – Sequential Access Device (SSC-3) . L – Printer Device (SSC) . P – Processor Device (SPC-2) . . W – Write Once Block Device (SBC) . . R – C/DVD Device (MMC-6) . . O – Optical Memory Block Device (SBC) . . . M – Media Changer Device (SMC-3) . . . A – Storage Array Device (SCC-2) . . . E – SCSI Enclosure Services device (SES) . . . . B – Simplified Direct-Access (Reduced Block) device (RBC) . . . . K – Optical Card Reader/Writer device (OCRW) . . . . V – Automation/Device Interface device (ADC) . . . . F – Object-based Storage Device (OSD)				<u>Device Column key</u> blank = code not used not blank = code used	
Log Page Code	Log Subpage Code	DTLPWROMAEBKVF	Log Page Name				
19h	08h	D	Group Statistics and Performance (8)				
19h	09h	D	Group Statistics and Performance (9)				
19h	0Ah	D	Group Statistics and Performance (10)				
19h	0Bh	D	Group Statistics and Performance (11)				
19h	0Ch	D	Group Statistics and Performance (12)				
19h	0Dh	D	Group Statistics and Performance (13)				
19h	0Eh	D	Group Statistics and Performance (14)				
19h	0Fh	D	Group Statistics and Performance (15)				
19h	10h	D	Group Statistics and Performance (16)				
19h	11h	D	Group Statistics and Performance (17)				
19h	12h	D	Group Statistics and Performance (18)				
19h	13h	D	Group Statistics and Performance (19)				
19h	14h	D	Group Statistics and Performance (20)				
19h	15h	D	Group Statistics and Performance (21)				
19h	16h	D	Group Statistics and Performance (22)				
19h	17h	D	Group Statistics and Performance (23)				
19h	18h	D	Group Statistics and Performance (24)				
19h	19h	D	Group Statistics and Performance (25)				
19h	1Ah	D	Group Statistics and Performance (26)				
19h	1Bh	D	Group Statistics and Performance (27)				
19h	1Ch	D	Group Statistics and Performance (28)				
19h	1Dh	D	Group Statistics and Performance (29)				
19h	1Eh	D	Group Statistics and Performance (30)				
19h	1Fh	D	Group Statistics and Performance (31)				
2Eh	00h	T	M	TapeAlert			
2Fh	00h	DTLPWROMAE	KV	Informational Exceptions			
30h to 3Eh	00h to FEh	Vendor specific					
3Fh	00h to FEh	Reserved					
All codes not shown here or in table D.11 are restricted or reserved.							

Table D.11 is a numerical order listing of the log page codes used by SCSI transport protocols.

**Table D.11 — Transport protocol specific log page codes**

Log Page Code	Log Subpage Code	F – Fibre Channel Protocol for SCSI (FCP-4) . S – Serial Attached SCSI (SAS-2) . V – Automation/Device Interface device (ADT-2) . U – USB Attached SCSI (UAS) . . . . .		Log Page Name	Device Column key blank = code not used not blank = code used
		FSVU			
18h	00h	S		Protocol Specific Port	
See table D.10 for information on the codes not shown here.					

## D.6 Mode page codes

Table D.12 is a numerical order listing of the mode page codes.

**Table D.12 — Mode page codes** (part 1 of 2)

		D – Direct Access Block Device (SBC-3) . T – Sequential Access Device (SSC-3) . L – Printer Device (SSC) . P – Processor Device (SPC-2) . W – Write Once Block Device (SBC) . R – C/DVD Device (MMC-6) . O – Optical Memory Block Device (SBC) . M – Media Changer Device (SMC-3) . A – Storage Array Device (SCC-2) . E – SCSI Enclosure Services device (SES) . B – Simplified Direct-Access (Reduced Block) device (RBC) . K – Optical Card Reader/Writer device (OCRW) . V – Automation/Device Interface device (ADC) . F – Object-based Storage Device (OSD)					Device Column key blank = code not used not blank = code used
Mode Page Code	Mode Subpage Code	DTLPWROMAEBKVF	Mode Page Name				
01h	00h	DT	WRO		K	Read-Write Error Recovery	
02h	00h	DTL	WROMAE		KVF	Disconnect-Reconnect	
03h	00h	D				Format Device	
03h	00h		L			Parallel Printer Interface	
03h	00h			R		MRW CD-RW	
04h	00h	D				Rigid Disk Geometry	
04h	00h		L			Serial Printer Interface	
05h	00h	D				Flexible Disk	
05h	00h		L			Printer Options	
05h	00h			R		Write Parameters	
06h	00h		W	O		Optical Memory	
06h	00h				B	RBC Device Parameters	
07h	00h	D	W	O	K	Verify Error Recovery	
08h	00h	D	WRO		K	Caching	
09h	00h	DTL	WRO	AE	K	obsolete	
0Ah	00h	DTL	WROMAE		KVF	Control	
0Ah	01h	DTL	WROMAE		KVF	Control Extension	
0Ah	F1h	D			B	Parallel ATA Control	
0Ah	F2h	D			B	Serial ATA Control	
0Bh	00h	D	W	O	K	Medium Types Supported	
0Ch	00h	D				Notch and Partition	
0Dh	00h	D				obsolete	
0Dh	00h			R		CD Device Parameters	
0Eh	00h			R		CD Audio Control	
0Eh	01h				V	Target Device	
0Eh	02h				V	DT Device Primary Port	
0Eh	03h				V	Logical Unit	
0Eh	04h				V	Target Device Serial Number	
0Fh	00h	T				Data Compression	
10h	00h	D				XOR Control	
10h	00h	T				Device Configuration	
11h	00h	T				Medium Partition (1)	
12h							
13h							

<sup>a</sup> MMC-4 calls this page the Fault/Failure Reporting mode page, however, the page format is a proper subset of the format described in 7.4.11 of this standard.

<sup>a</sup> MMC-4 calls this page the Fault/Failure Reporting mode page, however, the page format is a proper subset of the format described in 7.4.11 of this standard.

Table D.12 — Mode page codes (part 2 of 2)

		D – Direct Access Block Device (SBC-3) . T – Sequential Access Device (SSC-3) . L – Printer Device (SSC) . P – Processor Device (SPC-2) . W – Write Once Block Device (SBC) . R – C/DVD Device (MMC-6) . O – Optical Memory Block Device (SBC) . M – Media Changer Device (SMC-3) . A – Storage Array Device (SCC-2) . E – SCSI Enclosure Services device (SES) . B – Simplified Direct-Access (Reduced Block) device (RBC) . K – Optical Card Reader/Writer device (OCRW) . V – Automation/Device Interface device (ADC) . F – Object-based Storage Device (OSD)												Device Column key blank = code not used not blank = code used
Mode Page Code	Mode Subpage Code	DTLPWROMAEBKVF	Mode Page Name											
14h	00h	DT	P	W	R	O	M	A	E	B	K	V	F	Enclosure Services Management
15h														Extended
16h														Extended Device-Type Specific
17h														
18h	xxh	DTL	W	R	O	M	A	E				V	F	Protocol Specific Logical Unit (see table D.13)
19h	xxh	DTL	W	R	O	M	A	E				V	F	Protocol Specific Port (see table D.13)
1Ah	00h	DTL	W	R	O	M	A					V		Power Condition
1Ah	F1h	D										B		ATA Power Condition
1Bh	00h						A							LUN Mapping
1Ch	00h	DTL	W	R	O	M	A	E				V		Informational Exceptions Control <sup>a</sup>
1Ch	01h	D												Background Control
1Dh	00h					R								C/DVD Time-Out and Protect
1Dh	00h						M							Element Address Assignments
1Eh	00h						M							Transport Geometry Parameters
1Fh	00h						M							Device Capabilities
00h														Vendor specific (does not require page format)
20h to 29h														Device-type specific (vendor specific in common usage)
2Ah		DTL	W		O	M	A	E	B	K	V	F		Device-type specific (vendor specific in common usage)
2Ah	00h				R									CD Capabilities and Mechanical Status
2Bh to 3Eh														Device-type specific (vendor specific in common usage)
3Fh	xxh													Return all pages and/or subpages (MODE SENSE only)
xxh	FFh													Return all subpages (MODE SENSE only)
All codes not shown here or in table D.13 are restricted or reserved.														
<sup>a</sup> MMC-4 calls this page the Fault/Failure Reporting mode page, however, the page format is a proper subset of the format described in 7.4.11 of this standard.														

Table D.13 is a numerical order listing of the mode page codes used by SCSI transport protocols.

**Table D.13 — Transport protocol specific mode page codes**

Mode Page Code	Mode Subpage Code	F – Fibre Channel Protocol for SCSI (FCP-4) . S – Serial Attached SCSI (SAS-2) . V – Automation/Device Interface device (ADT-2) . U – USB Attached SCSI (UAS) . .		Mode Page Name
		FSVU		
18h	00h	FS		Protocol Specific Logical Unit
19h	00h	FS		Protocol Specific Port
19h	01h	S		Phy Control And Discover
19h	02h	S		Shared Port Control
See table D.12 for information on the codes not shown here.				

Device Column key  
 blank = code not used  
 not blank = code used

## D.7 VPD page codes

Table D.14 is a numerical order listing of the VPD page codes.

**Table D.14 — VPD page codes**

	D – Direct Access Block Device (SBC-3)				<u>Device Column key</u>	
	. T – Sequential Access Device (SSC-3)				blank = code not used	
	. L – Printer Device (SSC)				not blank = code used	
	. P – Processor Device (SPC-2)					
	. W – Write Once Block Device (SBC)					
	. R – C/DVD Device (MMC-6)					
	. O – Optical Memory Block Device (SBC)					
	. M – Media Changer Device (SMC-3)					
	. A – Storage Array Device (SCC-2)					
	. E – SCSI Enclosure Services device (SES)					
	. B – Simplified Direct-Access (Reduced Block) device (RBC)					
	. K – Optical Card Reader/Writer device (OCRW)					
	. V – Automation/Device Interface device (ADC)					
	. F – Object-based Storage Device (OSD)					
VPD Page Code						
	DTLPWROMAEBKVF	VPD Page Name				
00h	DTLPWROMAEBKVF	Supported VPD Pages				
01h to 7Fh	DTLPWROMAEBKVF	ASCII Information				
80h	DTLPWROMAEBKVF	Unit Serial Number				
82h	DTLPWROMAEBKVF	ASCII Implemented Operating Definition				
83h	DTLPWROMAEBKVF	Device Identification				
84h	DTLPWROMAEBKVF	Software Interface Identification				
85h	DTLPWROMAEBKVF	Management Network Addresses				
86h	DTLPWROMAEBKVF	Extended INQUIRY Data				
87h	DTLPWROMAEBKVF	Mode Page Policy				
88h	DTLPWROMAEBKVF	SCSI Ports				
89h	D	ATA Information				
90h	DTL WROMAE VF	Protocol Specific Logical Unit Information				
91h	DTL WROMAE VF	Protocol Specific Port Information				
B0h	D	Block Limits				
B0h	T	Sequential-access device capabilities				
B0h		F	OSD information			
B1h	D	Block Device Characteristics				
B1h	T	V	Manufacturer-assigned serial number			
B1h		F	Security token			
B2h	T	TapeAlert supported flags				
C0h to FFh		Vendor specific				
All codes not shown here or in table D.15 are restricted or reserved.						

Table D.13 is a numerical order listing of the VPD page codes used by SCSI transport protocols.

**Table D.15 — Transport protocol specific VPD page codes**

VPD Page Code	F – Fibre Channel Protocol for SCSI (FCP-4)		<u>Device Column key</u> blank = code not used not blank = code used
	. S – Serial Attached SCSI (SAS-2)		
	. V – Automation/Device Interface device (ADT-2)		
	. U – USB Attached SCSI (UAS)		
	FSVU	VPD Page Name	
90h	S	Protocol Specific Logical Unit Information	
91h	S	Protocol Specific Port Information	
See table D.14 for information on the codes not shown here.			

## D.8 Version descriptor values

Table D.16 is a numerical order listing of the version descriptor values used in the standard INQUIRY data. Each version descriptor value is computed from a coded value identifying the standard and a coded value representing the revision of the standard. The formula is  $((\text{standard} \times 32) + \text{revision})$ . Table D.16 shows all three code values and the associated standard name. The version descriptor code is shown in both decimal and hexadecimal.

**Table D.16 — Version descriptor assignments** (part 1 of 11)

Standard Code	Revision Code	Version Descriptor Code		Standard
		decimal	hex	
0	0	0	0000h	Version Descriptor Not Supported or No Standard Identified
1	0	32	0020h	SAM (no version claimed)
1	27	59	003Bh	SAM T10/0994-D revision 18
1	28	60	003Ch	SAM ANSI INCITS 270-1996
2	0	64	0040h	SAM-2 (no version claimed)
2	20	84	0054h	SAM-2 T10/1157-D revision 23
2	21	85	0055h	SAM-2 T10/1157-D revision 24
2	28	92	005Ch	SAM-2 ANSI INCITS 366-2003
2	30	94	005Eh	SAM-2 ISO/IEC 14776-412
3	0	96	0060h	SAM-3 (no version claimed)
3	2	98	0062h	SAM-3 T10/1561-D revision 7
3	21	117	0075h	SAM-3 T10/1561-D revision 13
3	22	118	0076h	SAM-3 T10/1561-D revision 14
3	23	119	0077h	SAM-3 ANSI INCITS 402-2005
4	0	128	0080h	SAM-4 (no version claimed)
4	7	135	0087h	SAM-4 T10/1683-D revision 13
4	11	139	008Bh	SAM-4 T10/1683-D revision 14
5	0	160	00A0h	SAM-5 (no version claimed)
9	0	288	0120h	SPC (no version claimed)
9	27	315	013Bh	SPC T10/0995-D revision 11a
9	28	316	013Ch	SPC ANSI INCITS 301-1997
10	0	320	0140h	MMC (no version claimed)
10	27	347	015Bh	MMC T10/1048-D revision 10a
10	28	348	015Ch	MMC ANSI INCITS 304-1997



Table D.16 — Version descriptor assignments (part 2 of 11)

Standard Code	Revision Code	Version Descriptor Code		Standard
		decimal	hex	
11	0	352	0160h	SCC (no version claimed)
11	27	379	017Bh	SCC T10/1047-D revision 06c
11	28	380	017Ch	SCC ANSI INCITS 276-1997
12	0	384	0180h	SBC (no version claimed)
12	27	411	019Bh	SBC T10/0996-D revision 08c
12	28	412	019Ch	SBC ANSI INCITS 306-1998
13	0	416	01A0h	SMC (no version claimed)
13	27	443	01BBh	SMC T10/0999-D revision 10a
13	28	444	01BCh	SMC ANSI INCITS 314-1998
13	30	446	01BEh	SMC ISO/IEC 14776-351
14	0	448	01C0h	SES (no version claimed)
14	27	475	01DBh	SES T10/1212-D revision 08b
14	28	476	01DCh	SES ANSI INCITS 305-1998
14	29	477	01DDh	SES T10/1212 revision 08b w/ Amendment ANSI INCITS.305/AM1-2000
14	30	478	01DEh	SES ANSI INCITS 305-1998 w/ Amendment ANSI INCITS.305/AM1-2000
15	0	480	01E0h	SCC-2 (no version claimed)
15	27	507	01FBh	SCC-2 T10/1125-D revision 04
15	28	508	01FCh	SCC-2 ANSI INCITS 318-1998
16	0	512	0200h	SSC (no version claimed)
16	1	513	0201h	SSC T10/0997-D revision 17
16	7	519	0207h	SSC T10/0997-D revision 22
16	28	540	021Ch	SSC ANSI INCITS 335-2000
17	0	544	0220h	RBC (no version claimed)
17	24	568	0238h	RBC T10/1240-D revision 10a
17	28	572	023Ch	RBC ANSI INCITS 330-2000
18	0	576	0240h	MMC-2 (no version claimed)
18	21	597	0255h	MMC-2 T10/1228-D revision 11
18	27	603	025Bh	MMC-2 T10/1228-D revision 11a

Table D.16 — Version descriptor assignments (part 3 of 11)

Standard Code	Revision Code	Version Descriptor Code		Standard
		decimal	hex	
18	28	604	025Ch	MMC-2 ANSI INCITS 333-2000
19	0	608	0260h	SPC-2 (no version claimed)
19	7	615	0267h	SPC-2 T10/1236-D revision 12
19	9	617	0269h	SPC-2 T10/1236-D revision 18
19	21	629	0275h	SPC-2 T10/1236-D revision 19
19	22	630	0276h	SPC-2 T10/1236-D revision 20
19	23	631	0277h	SPC-2 ANSI INCITS 351-2001
19	24	632	0278h	SPC-2 ISO/IEC 14776-452
20	0	640	0280h	OCRW (no version claimed)
20	30	670	029Eh	OCRW ISO/IEC 14776-381
21	0	672	02A0h	MMC-3 (no version claimed)
21	21	693	02B5h	MMC-3 T10/1363-D revision 9
21	22	694	02B6h	MMC-3 T10/1363-D revision 10g
21	24	696	02B8h	MMC-3 ANSI INCITS 360-2002
23	0	736	02E0h	SMC-2 (no version claimed)
23	21	757	02F5h	SMC-2 T10/1383-D revision 5
23	28	764	02FCh	SMC-2 T10/1383-D revision 6
23	29	765	02FDh	SMC-2 T10/1383-D revision 7
23	30	766	02FEh	SMC-2 ANSI INCITS 382-2004
24	0	768	0300h	SPC-3 (no version claimed)
24	1	769	0301h	SPC-3 T10/1416-D revision 7
24	7	775	0307h	SPC-3 T10/1416-D revision 21
24	15	783	030Fh	SPC-3 T10/1416-D revision 22
24	18	786	0312h	SPC-3 T10/1416-D revision 23
24	20	788	0314h	SPC-3 ANSI INCITS 408-2005
25	0	800	0320h	SBC-2 (no version claimed)
25	2	802	0322h	SBC-2 T10/1417-D revision 5a
25	4	804	0324h	SBC-2 T10/1417-D revision 15
25	27	827	033Bh	SBC-2 T10/1417-D revision 16

Table D.16 — Version descriptor assignments (part 4 of 11)

Standard Code	Revision Code	Version Descriptor Code		Standard
		decimal	hex	
25	29	829	033Dh	SBC-2 ANSI INCITS 405-2005
25	30	830	033Eh	SBC-2 ISO/IEC 14776-322
26	0	832	0340h	OSD (no version claimed)
26	1	833	0341h	OSD T10/1355-D revision 0
26	2	834	0342h	OSD T10/1355-D revision 7a
26	3	835	0343h	OSD T10/1355-D revision 8
26	4	836	0344h	OSD T10/1355-D revision 9
26	21	853	0355h	OSD T10/1355-D revision 10
26	22	854	0356h	OSD ANSI INCITS 400-2004
27	0	864	0360h	SSC-2 (no version claimed)
27	20	884	0374h	SSC-2 T10/1434-D revision 7
27	21	885	0375h	SSC-2 T10/1434-D revision 9
27	29	893	037Dh	SSC-2 ANSI INCITS 380-2003
28	0	896	0380h	BCC (no version claimed)
29	0	928	03A0h	MMC-4 (no version claimed)
29	16	944	03B0h	MMC-4 T10/1545-D revision 5
29	17	945	03B1h	MMC-4 T10/1545-D revision 5a
29	29	957	03BDh	MMC-4 T10/1545-D revision 3
29	30	958	03BEh	MMC-4 T10/1545-D revision 3d
29	31	959	03BFh	MMC-4 ANSI INCITS 401-2005
30	0	960	03C0h	ADC (no version claimed)
30	21	981	03D5h	ADC T10/1558-D revision 6
30	22	982	03D6h	ADC T10/1558-D revision 7
30	23	983	03D7h	ADC ANSI INCITS 403-2005
31	0	992	03E0h	SES-2 (no version claimed)
31	1	993	03E1h	SES-2 T10/1559-D revision 16
31	7	999	03E7h	SES-2 T10/1559-D revision 19
31	11	1003	03EBh	SES-2 T10/1559-D revision 20
32	0	1024	0400h	SSC-3 (no version claimed)

Table D.16 — Version descriptor assignments (part 5 of 11)

Standard Code	Revision Code	Version Descriptor Code		Standard
		decimal	hex	
32	3	1027	0403h	SSC-3 T10/1611-D revision 04a
33	0	1056	0420h	MMC-5 (no version claimed)
33	15	1071	042Fh	MMC-5 T10/1675-D revision 03
33	17	1073	0431h	MMC-5 T10/1675-D revision 03b
33	18	1074	0432h	MMC-5 T10/1675-D revision 04
33	20	1076	0434h	MMC-5 ANSI INCITS 430-2007
34	0	1088	0440h	OSD-2 (no version claimed)
34	4	1092	0444h	OSD-2 T10/1729-D revision 4
35	0	1120	0460h	SPC-4 (no version claimed)
35	1	1121	0461h	SPC-4 T10/1731-D revision 16
36	0	1152	0480h	SMC-3 (no version claimed)
37	0	1184	04A0h	ADC-2 (no version claimed)
37	7	1191	04A7h	ADC-2 T10/1741-D revision 7
37	10	1194	04AAh	ADC-2 T10/1741-D revision 8
37	12	1196	04ACh	ADC-2 ANSI INCITS 441-2008
38	0	1216	04C0h	SBC-3 (no version claimed)
39	0	1248	04E0h	MMC-6 (no version claimed)
40	0	1280	0500h	ADC-3 (no version claimed)
65	0	2080	0820h	SSA-TL2 (no version claimed)
65	27	2107	083Bh	SSA-TL2 T10.1/1147-D revision 05b
65	28	2108	083Ch	SSA-TL2 ANSI INCITS 308-1998
66	0	2112	0840h	SSA-TL1 (no version claimed)
66	27	2139	085Bh	SSA-TL1 T10.1/0989-D revision 10b
66	28	2140	085Ch	SSA-TL1 ANSI INCITS 295-1996
67	0	2144	0860h	SSA-S3P (no version claimed)
67	27	2171	087Bh	SSA-S3P T10.1/1051-D revision 05b
67	28	2172	087Ch	SSA-S3P ANSI INCITS 309-1998
68	0	2176	0880h	SSA-S2P (no version claimed)
68	27	2203	089Bh	SSA-S2P T10.1/1121-D revision 07b

Table D.16 — Version descriptor assignments (part 6 of 11)

Standard Code	Revision Code	Version Descriptor Code		Standard
		decimal	hex	
68	28	2204	089Ch	SSA-S2P ANSI INCITS 294-1996
69	0	2208	08A0h	SIP (no version claimed)
69	27	2235	08BBh	SIP T10/0856-D revision 10
69	28	2236	08BCh	SIP ANSI INCITS 292-1997
70	0	2240	08C0h	FCP (no version claimed)
70	27	2267	08DBh	FCP T10/0993-D revision 12
70	28	2268	08DCh	FCP ANSI INCITS 269-1996
71	0	2272	08E0h	SBP-2 (no version claimed)
71	27	2299	08FBh	SBP-2 T10/1155-D revision 04
71	28	2300	08FCh	SBP-2 ANSI INCITS 325-1998
72	0	2304	0900h	FCP-2 (no version claimed)
72	1	2305	0901h	FCP-2 T10/1144-D revision 4
72	21	2325	0915h	FCP-2 T10/1144-D revision 7
72	22	2326	0916h	FCP-2 T10/1144-D revision 7a
72	23	2327	0917h	FCP-2 ANSI INCITS 350-2003
72	24	2328	0918h	FCP-2 T10/1144-D revision 8
73	0	2336	0920h	SST (no version claimed)
73	21	2357	0935h	SST T10/1380-D revision 8b
74	0	2368	0940h	SRP (no version claimed)
74	20	2388	0954h	SRP T10/1415-D revision 10
74	21	2389	0955h	SRP T10/1415-D revision 16a
74	28	2396	095Ch	SRP ANSI INCITS 365-2002
75	0	2400	0960h	iSCSI (no version claimed)
76	0	2432	0980h	SBP-3 (no version claimed)
76	2	2434	0982h	SBP-3 T10/1467-D revision 1f
76	20	2452	0994h	SBP-3 T10/1467-D revision 3
76	26	2458	099Ah	SBP-3 T10/1467-D revision 4
76	27	2459	099Bh	SBP-3 T10/1467-D revision 5
76	28	2460	099Ch	SBP-3 ANSI INCITS 375-2004

Table D.16 — Version descriptor assignments (part 7 of 11)

Standard Code	Revision Code	Version Descriptor Code		Standard
		decimal	hex	
78	0	2496	09C0h	ADP (no version claimed)
79	0	2528	09E0h	ADT (no version claimed)
79	25	2553	09F9h	ADT T10/1557-D revision 11
79	26	2554	09FAh	ADT T10/1557-D revision 14
79	29	2557	09FDh	ADT ANSI INCITS 406-2005
80	0	2560	0A00h	FCP-3 (no version claimed)
80	7	2567	0A07h	FCP-3 T10/1560-D revision 3f
80	15	2575	0A0Fh	FCP-3 T10/1560-D revision 4
80	17	2577	0A11h	FCP-3 ANSI INCITS 416-2006
80	28	2588	0A1Ch	FCP-3 ISO/IEC 14776-223
81	0	2592	0A20h	ADT-2 (no version claimed)
82	0	2624	0A40h	FCP-4 (no version claimed)
85	0	2720	0AA0h	SPI (no version claimed)
85	25	2745	0AB9h	SPI T10/0855-D revision 15a
85	26	2746	0ABAh	SPI ANSI INCITS 253-1995
85	27	2747	0AB Bh	SPI T10/0855-D revision 15a with SPI Amnd revision 3a
85	28	2748	0ABCh	SPI ANSI INCITS 253-1995 with SPI Amnd ANSI INCITS 253/AM1-1998
86	0	2752	0AC0h	Fast-20 (no version claimed)
86	27	2779	0ADBh	Fast-20 T10/1071 revision 06
86	28	2780	0ADCh	Fast-20 ANSI INCITS 277-1996
87	0	2784	0AE0h	SPI-2 (no version claimed)
87	27	2811	0AF Bh	SPI-2 T10/1142-D revision 20b
87	28	2812	0AFCh	SPI-2 ANSI INCITS 302-1999
88	0	2816	0B00h	SPI-3 (no version claimed)
88	24	2840	0B18h	SPI-3 T10/1302-D revision 10
88	25	2841	0B19h	SPI-3 T10/1302-D revision 13a
88	26	2842	0B1Ah	SPI-3 T10/1302-D revision 14
88	28	2844	0B1Ch	SPI-3 ANSI INCITS 336-2000

Table D.16 — Version descriptor assignments (part 8 of 11)

Standard Code	Revision Code	Version Descriptor Code		Standard
		decimal	hex	
89	0	2848	0B20h	EPI (no version claimed)
89	27	2875	0B3Bh	EPI T10/1134 revision 16
89	28	2876	0B3Ch	EPI ANSI INCITS TR-23 1999
90	0	2880	0B40h	SPI-4 (no version claimed)
90	20	2900	0B54h	SPI-4 T10/1365-D revision 7
90	21	2901	0B55h	SPI-4 T10/1365-D revision 9
90	22	2902	0B56h	SPI-4 ANSI INCITS 362-2002
90	25	2905	0B59h	SPI-4 T10/1365-D revision 10
91	0	2912	0B60h	SPI-5 (no version claimed)
91	25	2937	0B79h	SPI-5 T10/1525-D revision 3
91	26	2938	0B7Ah	SPI-5 T10/1525-D revision 5
91	27	2939	0B7Bh	SPI-5 T10/1525-D revision 6
91	28	2940	0B7Ch	SPI-5 ANSI INCITS 367-2003
95	0	3040	0BE0h	SAS (no version claimed)
95	1	3041	0BE1h	SAS T10/1562-D revision 01
95	21	3061	0BF5h	SAS T10/1562-D revision 03
95	26	3066	0BFAh	SAS T10/1562-D revision 04
95	27	3067	0BFBh	SAS T10/1562-D revision 04
95	28	3068	0BFCh	SAS T10/1562-D revision 05
95	29	3069	0BFDh	SAS ANSI INCITS 376-2003
96	0	3072	0C00h	SAS-1.1 (no version claimed)
96	7	3079	0C07h	SAS-1.1 T10/1601-D revision 9
96	15	3087	0C0Fh	SAS-1.1 T10/1601-D revision 10
96	17	3089	0C11h	SAS-1.1 ANSI INCITS 417-2006
97	0	3104	0C20h	SAS-2 (no version claimed)
97	3	3107	0C23h	SAS-2 T10/1760-D revision 14
105	0	3360	0D20h	FC-PH (no version claimed)
105	27	3387	0D3Bh	FC-PH ANSI INCITS 230-1994

Table D.16 — Version descriptor assignments (part 9 of 11)

Standard Code	Revision Code	Version Descriptor Code		Standard
		decimal	hex	
105	28	3388	0D3Ch	FC-PH ANSI INCITS 230-1994 with Amnd 1 ANSI INCITS 230/AM1-1996
106	0	3392	0D40h	FC-AL (no version claimed)
106	28	3420	0D5Ch	FC-AL ANSI INCITS 272-1996
107	0	3424	0D60h	FC-AL-2 (no version claimed)
107	1	3425	0D61h	FC-AL-2 T11/1133-D revision 7.0
107	3	3427	0D63h	FC-AL-2 ANSI INCITS 332-1999 with AM1-2003 & AM2-2006
107	4	3428	0D64h	FC-AL-2 ANSI INCITS 332-1999 with Amnd 2 AM2-2006
107	28	3452	0D7Ch	FC-AL-2 ANSI INCITS 332-1999
107	29	3453	0D7Dh	FC-AL-2 ANSI INCITS 332-1999 with Amnd 1 AM1-2003
108	0	3456	0D80h	FC-PH-3 (no version claimed)
108	28	3484	0D9Ch	FC-PH-3 ANSI INCITS 303-1998
109	0	3488	0DA0h	FC-FS (no version claimed)
109	23	3511	0DB7h	FC-FS T11/1331-D revision 1.2
109	24	3512	0DB8h	FC-FS T11/1331-D revision 1.7
109	28	3516	0DBCh	FC-FS ANSI INCITS 373-2003
110	0	3520	0DC0h	FC-PI (no version claimed)
110	28	3548	0DDCh	FC-PI ANSI INCITS 352-2002
111	0	3552	0DE0h	FC-PI-2 (no version claimed)
111	2	3554	0DE2h	FC-PI-2 T11/1506-D revision 5.0
111	4	3556	0DE4h	FC-PI-2 ANSI INCITS 404-2006
112	0	3584	0E00h	FC-FS-2 (no version claimed)
112	2	3586	0E02h	FC-FS-2 ANSI INCITS 242-2007
112	3	3587	0E03h	FC-FS-2 ANSI INCITS 242-2007 with AM1 ANSI INCITS 242/AM1-2007
113	0	3616	0E20h	FC-LS (no version claimed)
113	1	3617	0E21h	FC-LS T11/1620-D revision 1.62
113	9	3625	0E29h	FC-LS ANSI INCITS 433-2007
114	0	3648	0E40h	FC-SP (no version claimed)
114	2	3650	0E42h	FC-SP T11/1570-D revision 1.6



Table D.16 — Version descriptor assignments (part 10 of 11)

Standard Code	Revision Code	Version Descriptor Code		Standard
		decimal	hex	
114	5	3653	0E45h	FC-SP ANSI INCITS 426-2007
115	0	3680	0E60h	FC-PI-3 (no version claimed)
116	0	3712	0E80h	FC-PI-4 (no version claimed)
116	2	3714	0E82h	FC-PI-4 T11/1647-D revision 8.0
117	0	3744	0EA0h	FC 10GFC (no version claimed)
117	2	3746	0EA2h	FC 10GFC ANSI INCITS 364-2003
117	3	3747	0EA3h	FC 10GFC ISO/IEC 14165-116
117	6	3750	0EA6h	FC 10GFC ANSI INCITS 364-2003 with AM1 ANSI INCITS 364/AM1-2007
118	0	3776	0EC0h	FC-SP-2 (no version claimed)
119	0	3808	0EE0h	FC-FS-3 (no version claimed)
120	0	3840	0F00h	FC-LS-2 (no version claimed)
150	0	4800	12C0h	FC-DA-2 (no version claimed)
151	0	4832	12E0h	FC-DA (no version claimed)
151	2	4834	12E2h	FC-DA T11/1513-DT revision 3.1
151	8	4840	12E8h	FC-DA ANSI INCITS TR-36 2004
152	0	4864	1300h	FC-Tape (no version claimed)
152	1	4865	1301h	FC-Tape T11/1315 revision 1.16
152	27	4891	131Bh	FC-Tape T11/1315 revision 1.17
152	28	4892	131Ch	FC-Tape ANSI INCITS TR-24 1999
153	0	4896	1320h	FC-FLA (no version claimed)
153	27	4923	133Bh	FC-FLA T11/1235 revision 7
153	28	4924	133Ch	FC-FLA ANSI INCITS TR-20 1998
154	0	4928	1340h	FC-PLDA (no version claimed)
154	27	4955	135Bh	FC-PLDA T11/1162 revision 2.1
154	28	4956	135Ch	FC-PLDA ANSI INCITS TR-19 1998
155	0	4960	1360h	SSA-PH2 (no version claimed)
155	27	4987	137Bh	SSA-PH2 T10.1/1145-D revision 09c
155	28	4988	137Ch	SSA-PH2 ANSI INCITS 293-1996

Table D.16 — Version descriptor assignments (part 11 of 11)

Standard Code	Revision Code	Version Descriptor Code		Standard
		decimal	hex	
156	0	4992	1380h	SSA-PH3 (no version claimed)
156	27	5019	139Bh	SSA-PH3 T10.1/1146-D revision 05b
156	28	5020	139Ch	SSA-PH3 ANSI INCITS 307-1998
165	0	5280	14A0h	IEEE 1394 (no version claimed)
165	29	5309	14BDh	ANSI IEEE 1394-1995
166	0	5312	14C0h	IEEE 1394a (no version claimed)
167	0	5344	14E0h	IEEE 1394b (no version claimed)
175	0	5600	15E0h	ATA/ATAPI-6 (no version claimed)
175	29	5629	15FDh	ATA/ATAPI-6 ANSI INCITS 361-2002
176	0	5632	1600h	ATA/ATAPI-7 (no version claimed)
176	2	5634	1602h	ATA/ATAPI-7 T13/1532-D revision 3
176	28	5660	161Ch	ATA/ATAPI-7 ANSI INCITS 397-2005
177	0	5664	1620h	ATA/ATAPI-8 ATA8-AAM Architecture Model (no version claimed)
177	1	5665	1621h	ATA/ATAPI-8 ATA8-APT Parallel Transport (no version claimed)
177	2	5666	1622h	ATA/ATAPI-8 ATA8-AST Serial Transport (no version claimed)
177	3	5667	1623h	ATA/ATAPI-8 ATA8-ACS ATA/ATAPI Command Set (no version claimed)
185	8	5928	1728h	Universal Serial Bus Specification, Revision 1.1
185	9	5929	1729h	Universal Serial Bus Specification, Revision 2.0
185	16	5936	1730h	USB Mass Storage Class Bulk-Only Transport, Revision 1.0
186	0	5952	1740h	UAS (no version claimed)
245	0	7840	1EA0h	SAT (no version claimed)
245	7	7847	1EA7h	SAT T10/1711-D revision 8
245	11	7851	1EABh	SAT T10/1711-D revision 9
245	13	7853	1EADh	SAT ANSI INCITS 431-2007
246	0	7872	1EC0h	SAT-2 (no version claimed)
246	4	7876	1EC4h	SAT-2 T10/1826-D revision 6

Table D.17 shows the guidelines used by T10 when selecting a coded value for a standard.

**Table D.17 — Standard code value guidelines (part 1 of 4)**

<b>Standard Code</b>	<b>Standard or standards family</b>
0	Version Descriptor Not Supported
1 to 8	Architecture Model
1	SAM
2	SAM-2
3	SAM-3
4	SAM-4
5	SAM-5
9 to 64	Command Set
9	SPC
10	MMC
11	SCC
12	SBC
13	SMC
14	SES
15	SCC-2
16	SSC
17	RBC
18	MMC-2
19	SPC-2
20	OCRW
21	MMC-3
22	obsolete
23	SMC-2
24	SPC-3
25	SBC-2
26	OSD
27	SSC-2
28	BCC
29	MMC-4
30	ADC

Table D.17 — Standard code value guidelines (part 2 of 4)

Standard Code	Standard or standards family
31	SES-2
32	SSC-3
33	MMC-5
34	OSD-2
35	SPC-4
36	SMC-3
37	ADC-2
38	SBC-3
39	MMC-6
40	ADC-3
65 to 84	Physical Mapping Protocol
65	SSA-TL2
66	SSA-TL1
67	SSA-S3P
68	SSA-S2P
69	SIP
70	FCP
71	SBP-2
72	FCP-2
73	SST
74	SRP
75	iSCSI
76	SBP-3
77	obsolete
78	ADP
79	ADT
80	FCP-3
81	ADT-2
82	FCP-4
85 to 94	Parallel SCSI Physical
85	SPI and SPI Amendment

Table D.17 — Standard code value guidelines (part 3 of 4)

Standard Code	Standard or standards family
86	Fast-20
87	SPI-2
88	SPI-3
89	EPI
90	SPI-4
91	SPI-5
95 to 104	Serial Attached SCSI
95	SAS
96	SAS-1.1
97	SAS-2
105 to 154	Fibre Channel
105	FC-PH and FC-PH Amendment
106	FC-AL
107	FC-AL-2
108	FC-PH-3
109	FC-FS
110	FC-PI
111	FC-PI-2
112	FC-FS-2
113	FC-LS
114	FC-SP
115	FC-PI-3
116	FC-PI-4
117	FC 10GFC
118	FC-SP-2
119	FC-FS-3
120	FC-LS-2
150	FC-DA-2
151	FC-DA
152	FC-Tape
153	FC-FLA

**Table D.17 — Standard code value guidelines** (part 4 of 4)

<b>Standard Code</b>	<b>Standard or standards family</b>
154	FC-PLDA
155 to 164	SSA
155	SSA-PH2
156	SSA-PH3
165 to 174	IEEE 1394
165	IEEE 1394:1995
166	IEEE 1394a
167	IEEE 1394b
175 to 200	ATAPI & USB
175	ATA/ATAPI-6
176	ATA/ATAPI-7
177	ATA/ATAPI-8
185	USB
186	UAS
201 to 224	Networking
225 to 244	ATM
245 to 264	Translators
245	SAT
246	SAT-2
265 to 2047	Reserved for Expansion

Table D.18 shows the guidelines used by T10 when selecting a coded value for a revision.

**Table D.18 — Revision code value guidelines**

<b>Revision Code</b>	<b>Revision</b>
0	No revision specified
1 to 20	Revisions accepted as T10 committee drafts or stabilized drafts
20 to 29	Revisions for letter ballot, forwarded to INCITS or ANSI approved
30 to 32	Revisions approved as international standards

## D.9 T10 IEEE binary identifiers

The IEEE binary identifiers assigned to T10 standards are shown in table D.19.

**Table D.19 — IEEE binary identifiers assigned by T10**

IEEE Binary Identifier	T10 Standard
0060 9E01 03E0h	SBP (obsolete)
0060 9E01 0483h	SBP-2
0060 9E01 04D8h	SPC-2
0060 9E01 05BBh	SBP-3
0060 9E01 0800h	SRP revision 10
0060 9E01 0801h	ANSI SRP

## Annex E

(informative)

### T10 vendor identification

This annex contains the list of T10 vendor identifications (see table E.1) as of the date of this document. The purpose of this list is to help avoid redundant usage of T10 vendor identifications. Technical Committee T10 of Accredited Standards Committee INCITS maintains an informal list of T10 vendor identifications currently in use. The T10 web site, <http://www.t10.org>, provides a convenient means to request an identification code. If problems are encountered using the T10 web site, please contact the chairman of T10 prior to using a new T10 vendor identification to avoid conflicts.

The information in this annex was complete and accurate at the time of publication. However, the information is subject to change. Technical Committee T10 of INCITS maintains an electronic copy of this information on its world wide web site (<http://www.t10.org/>). In the event that the T10 world wide web site is no longer active, access may be possible via the INCITS world wide web site (<http://www.incits.org>), the ANSI world wide web site (<http://www.ansi.org>), the IEC site (<http://www.iec.ch/>), the ISO site (<http://www.iso.ch/>), or the ISO/IEC JTC 1 web site (<http://www.jtc1.org/>).

**Table E.1 — T10 vendor identification list (part 1 of 13)**

ID	Organization
0B4C	MOOSIK Ltd.
2AI	2AI (Automatisme et Avenir Informatique)
3M	3M Company
3PARdata	3PARdata, Inc.
A-Max	A-Max Technology Co., Ltd
ACARD	ACARD Technology Corp.
Accusys	Accusys INC.
Acer	Acer, Inc.
ACL	Automated Cartridge Librarys, Inc.
Acuid	Acuid Corporation Ltd.
AcuLab	AcuLab, Inc. (Tulsa, OK)
ADAPTEC	Adaptec
ADIC	Advanced Digital Information Corporation
ADSI	Adaptive Data Systems, Inc. (a Western Digital subsidiary)
ADTX	ADTX Co., Ltd.
ADVA	ADVA Optical Networking AG
AERONICS	Aeronics, Inc.
AGFA	AGFA
AIPTEK	AIPTEK International Inc.
AMCC	Applied Micro Circuits Corporation
AMCODYNE	Amcodyne
Amgeon	Amgeon LLC
AMI	American Megatrends, Inc.
Amphenol	Amphenol
Amtl	Tenlon Technology Co.,Ltd
ANAMATIC	Anamartic Limited (England)
Ancor	Ancor Communications, Inc.
ANCOT	ANCOT Corp.



**Table E.1 — T10 vendor identification list** (part 2 of 13)

<b>ID</b>	<b>Organization</b>
ANDATACO	Andataco (now nStor)
andiamo	Andiamo Systems, Inc.
ANRITSU	Anritsu Corporation
ANTONIO	Antonio Precise Products Manufactory Ltd.
APPLE	Apple Computer, Inc.
ARCHIVE	Archive
ARDENCE	Ardence Inc
ARIO	Ario Data Networks, Inc.
ARISTOS	Aristos Logic Corp.
ARK	ARK Research Corporation
ARTECON	Artecon Inc. (Obs. - now Dot Hill)
ASACA	ASACA Corp.
ASC	Advanced Storage Concepts, Inc.
ASPEN	Aspen Peripherals
AST	AST Research
ASTK	Alcatel STK A/S
ASTUTE	Astute Networks, Inc.
AT&T	AT&T
ATA	SCSI / ATA Translator Software (Organization Not Specified)
ATARI	Atari Corporation
ATG CYG	ATG Cygnet Inc.
ATL	Quantum/ATL Products
ATTO	ATTO Technology Inc.
ATX	Alphatronix
AVC	AVC Technology Ltd
AVR	Advanced Vision Research
AXSTOR	AXSTOR
BALLARD	Ballard Synergy Corp.
Barco	Barco
BAROMTEC	Barom Technologies Co., Ltd.
BDT	BDT AG
BECEEM	Beceem Communications, Inc
BENQ	BENQ Corporation.
BERGSWD	Berg Software Design
BEZIER	Bezier Systems, Inc.
BHTi	Breece Hill Technologies
BIOS	BIOS Corporation
BIR	Bio-Imaging Research, Inc.
BiT	BiT Microsystems (obsolete, new ID: BITMICRO)
BITMICRO	BiT Microsystems, Inc.
BLOOMBAS	Bloombase Technologies Limited
BlueArc	BlueArc Corporation
BNCHMARK	Benchmark Tape Systems Corporation
Bosch	Robert Bosch GmbH
BoxHill	Box Hill Systems Corporation (Obs. - now Dot Hill)
BREA	BREA Technologies, Inc.
BREECE	Breece Hill LLC

**Table E.1 — T10 vendor identification list** (part 3 of 13)

<b>ID</b>	<b>Organization</b>
Broadcom	Broadcom Corporation
BROCADE	Brocade Communications Systems, Incorporated
BUFFALO	BUFFALO INC.
BULL	Bull Peripherals Corp.
BUSLOGIC	BusLogic Inc.
CalComp	CalComp, A Lockheed Company
CALIPER	Caliper (California Peripheral Corp.)
CAMBEX	Cambex Corporation
CAMEOSYS	Cameo Systems Inc.
CANDERA	Candera Inc.
CAPTION	CAPTION BANK
CAST	Advanced Storage Tech
CATALYST	Catalyst Enterprises
CCDISK	iSCSI Cake
CDC	Control Data or MPI
CDP	Columbia Data Products
CenData	Central Data Corporation
Cereva	Cereva Networks Inc.
CERTANCE	Certance
CHEROKEE	Cherokee Data Systems
CHINON	Chinon
CIE&YED	YE Data, C.Itoh Electric Corp.
CIPHER	Cipher Data Products
Ciprico	Ciprico, Inc.
CIRRUSL	Cirrus Logic Inc.
CISCO	Cisco Systems, Inc.
CLOVERLF	Cloverleaf Communications, Inc
CMD	CMD Technology Inc.
CMTechno	CMTech
CNGR SFW	Congruent Software, Inc.
CNSi	Chaparral Network Storage, Inc.
CNT	Computer Network Technology
COBY	Coby Electronics Corporation, USA
COGITO	Cogito
COMPAQ	Compaq Computer Corporation (now HP)
COMPELNT	Compellent Technologies, Inc.
COMPORT	Comport Corp.
COMPSIG	Computer Signal Corporation
COMPTX	Comptex Pty Limited
CONNER	Conner Peripherals
COPANSYS	COPAN SYSTEMS INC
CORE	Core International, Inc.
COVOTE	Covote GmbH & Co KG
COWON	COWON SYSTEMS, Inc.
CPL	Cross Products Ltd
CPU TECH	CPU Technology, Inc.
CREO	Creo Products Inc.

**Table E.1 — T10 vendor identification list** (part 4 of 13)

<b>ID</b>	<b>Organization</b>
CROSFELD	Crosfield Electronics (now FujiFilm Electronic Imaging Ltd)
CROSSRDS	Crossroads Systems, Inc.
crosswlk	Crosswalk, Inc.
CSCOVRTS	Cisco - Veritas
CSM, INC	Computer SM, Inc.
CYBERNET	Cybernetics
Cygnal	Dekimo
DALSEMI	Dallas Semiconductor
DANEELEC	Dane-Elec
DANGER	Danger Inc.
DAT-MG	DAT Manufacturers Group
Data Com	Data Com Information Systems Pty. Ltd.
DATABOOK	Databook, Inc.
DATAcopy	Datacopy Corp.
DataCore	DataCore Software Corporation
DATAPT	Datapoint Corp.
DDN	DataDirect Networks, Inc.
DDRDRIVE	DDRdrive LLC
DEC	Digital Equipment Corporation (now HP)
DEI	Digital Engineering, Inc.
DELL	Dell Computer Corporation
DELPHI	Delphi Data Div. of Sparks Industries, Inc.
DENON	Denon/Nippon Columbia
DenOptix	DenOptix, Inc.
DEST	DEST Corp.
DGC	Data General Corp.
DIGIDATA	Digi-Data Corporation
DigiIntl	Digi International
Digital	Digital Equipment Corporation (now HP)
DILOG	Distributed Logic Corp.
DISC	Document Imaging Systems Corp.
DLNET	Driveline
DNS	Data and Network Security
DotHill	Dot Hill Systems Corp.
DP	Dell, Inc.
DPT	Distributed Processing Technology
DROBO	Data Robotics, Inc.
DSC	DigitalStream Corporation
DSI	Data Spectrum, Inc.
DSM	Deterner Steuerungs- und Maschinenbau GmbH & Co.
DSNET	Cleversafe, Inc.
DTC QUME	Data Technology Qume
DXIMAGIN	DX Imaging
ECCS	ECCS, Inc.
ECMA	European Computer Manufacturers Association
elipsan	Elipsan UK Ltd.
Elms	Elms Systems Corporation

**Table E.1 — T10 vendor identification list** (part 5 of 13)

<b>ID</b>	<b>Organization</b>
EMASS	EMASS, Inc.
EMC	EMC Corp.
EMTEC	EMTEC Magnetics
EMULEX	Emulex
ENERGY-B	Energybeam Corporation
ENGENIO	Engenio Information Technologies, Inc.
EPOS	EPOS Technologies Ltd.
EPSON	Epson
EQLOGIC	EqualLogic
Eris/RSI	RSI Systems, Inc.
EuroLogic	Eurologic Systems Limited
evolve	Evolution Technologies, Inc
EXABYTE	Exabyte Corp.
EXATEL	Exatelecom Co., Ltd.
EXAVIO	Exavio, Inc.
Exsequi	Exsequi Ltd
FALCON	FalconStor, Inc.
FFEILTD	FujiFilm Electronic Imaging Ltd
Fibxn	Fiberxon, Inc.
FID	First International Digital, Inc.
FILENET	FileNet Corp.
FirmFact	Firmware Factory Ltd
FRAMDRV	FRAMEDRIVE Corp.
FREECION	Nable Communications, Inc.
FSC	Fujitsu Siemens Computers
FTPL	Frontline Technologies Pte Ltd
FUJI	Fuji Electric Co., Ltd. (Japan)
FUJIFILM	Fuji Photo Film, Co., Ltd.
FUJITSU	Fujitsu
FUNAI	Funai Electric Co., Ltd.
FUTURED	Future Domain Corp.
G&D	Giesecke & Devrient GmbH
Gadzoox	Gadzoox Networks, Inc.
Gammaflx	Gammaflux L.P.
GDI	Generic Distribution International
Gen_Dyn	General Dynamics
Generic	Generic Technology Co., Ltd.
GENSIG	General Signal Networks
GIGATAPE	GIGATAPE GmbH
GIGATRND	GigaTrend Incorporated
Global	Global Memory Test Consortium
Gnutek	Gnutek Ltd.
Goidelic	Goidelic Precision, Inc.
GoldStar	LG Electronics Inc.
GOULD	Gould
HAGIWARA	Hagiwara Sys-Com Co., Ltd.
HAPP3	Inventec Multimedia and Telecom co., Ltd

**Table E.1 — T10 vendor identification list** (part 6 of 13)

<b>ID</b>	<b>Organization</b>
HDS	Horizon Data Systems, Inc.
Heydays	Mazo Technology Co., Ltd.
HI-TECH	HI-TECH Software Pty. Ltd.
HITACHI	Hitachi America Ltd or Nissei Sangyo America Ltd
HL-DT-ST	Hitachi-LG Data Storage, Inc.
HONEYWEL	Honeywell Inc.
HP	Hewlett Packard
HPQ	Hewlett Packard
HYUNWON	HYUNWON inc
i-cubed	i-cubed Ltd.
IBM	International Business Machines
ICL	ICL
ICP	ICP vortex Computersysteme GmbH
IDE	International Data Engineering, Inc.
IDG	Interface Design Group
IET	ISCSI ENTERPRISE TARGET
IFT	Infortrend Technology, Inc.
IGR	Intergraph Corp.
IMATION	Imation
IMPLTD	Integrated Micro Products Ltd.
IMPRIMIS	Imprimis Technology Inc.
INCIPNT	Incipient Technologies Inc.
INCITS	InterNational Committee for Information Technology
INDCOMP	Industrial Computing Limited
Indigita	Indigita Corporation
INITIO	Initio Corporation
INRANGE	INRANGE Technologies Corporation
INSITE	Insite Peripherals
integrix	Integrix, Inc.
INTEL	Intel Corporation
Intransa	Intransa, Inc.
IOC	I/O Concepts, Inc.
iofy	iofy Corporation
IOMEGA	Iomega
iqstor	iQstor Networks, Inc.
iQue	iQue
ISi	Information Storage inc.
Isilon	Isilon Systems, Inc.
ISO	International Standards Organization
iStor	iStor Networks, Inc.
ITC	International Tapetronics Corporation
IVIVITY	iVivity, Inc.
IVMMLTD	InnoVISION Multimedia Ltd.
JETWAY	Jetway Information Co., Ltd
JPC Inc.	JPC Inc.
JSCSI	jSCSI Project
JVC	JVC Information Products Co.

**Table E.1 — T10 vendor identification list** (part 7 of 13)

<b>ID</b>	<b>Organization</b>
KASHYA	Kashya, Inc.
KENNEDY	Kennedy Company
KENWOOD	KENWOOD Corporation
KEWL	Shanghai KEWL Imp&Exp Co., Ltd.
KODAK	Eastman Kodak
KONAN	Konan
koncepts	koncepts International Ltd.
KONICA	Konica Japan
KSCOM	KSCOM Co. Ltd.,
KUDELSKI	Nagravision SA - Kudelski Group
Kyocera	Kyocera Corporation
LAPINE	Lapine Technology
LASERDRV	LaserDrive Limited
LASERGR	Lasergraphics, Inc.
LeapFrog	LeapFrog Enterprises, Inc.
LEFTHAND	LeftHand Networks
Lexar	Lexar Media, Inc.
LG	LG Electronics Inc.
LGE	LG Electronics Inc.
LION	Lion Optics Corporation
LMS	Laser Magnetic Storage International Company
LSI	LSI Corp. (was LSI Logic Corp.)
LSILOGIC	LSI Logic Storage Systems, Inc.
LTO-CVE	Linear Tape - Open, Compliance Verification Entity
LUXPRO	Luxpro Corporation
Malakite	Malachite Technologies (New VID is: Sandial)
Marner	Marner Storage Technologies, Inc.
MARVELL	Marvell Semiconductor, Inc.
MATSHITA	Matsushita
MAXELL	Hitachi Maxell, Ltd.
MAXIM-IC	Maxim Integrated Products
MaxOptix	Maxoptix Corp.
MAXSTRAT	Maximum Strategy, Inc.
MAXTOR	Maxtor Corp.
MaXXan	MaXXan Systems, Inc.
MAYCOM	maycom Co., Ltd.
MBEAT	K-WON C&C Co.,Ltd
MCC	Measurement Computing Corporation
McDATA	McDATA Corporation
MDI	Micro Design International, Inc.
MEADE	Meade Instruments Corporation
MEII	Mountain Engineering II, Inc.
MELA	Mitsubishi Electronics America
MELCO	Mitsubishi Electric (Japan)
MEMOREX	Memorex Telex Japan Ltd.
MEMREL	Memrel Corporation
MEMTECH	MemTech Technology

**Table E.1 — T10 vendor identification list** (part 8 of 13)

<b>ID</b>	<b>Organization</b>
MendoCno	Mendocino Software
MERIDATA	Oy Meridata Finland Ltd
METRUM	Metrum, Inc.
MHTL	Matsunichi Hi-Tech Limited
MICROBTX	Microbotics Inc.
Microchp	Microchip Technology, Inc.
MICROLIT	Microlite Corporation
MICROP	Micropolis
MICROTEK	Microtek Storage Corp
Minitech	Minitech (UK) Limited
Minolta	Minolta Corporation
MINScriB	Miniscribe
MiraLink	MiraLink Corporation
MITSUMI	Mitsumi Electric Co., Ltd.
MKM	Mitsubishi Kagaku Media Co., LTD.
MOSAID	Mosaid Technologies Inc.
MOTOROLA	Motorola
MP-400	Daiwa Manufacturing Limited
MPC	MPC Corporation
MPCCORP	MPC Computers
MPEYE	Touchstone Technology Co., Ltd
MPIO	DKT Co.,Ltd
MPM	Mitsubishi Paper Mills, Ltd.
MPMan	MPMan.com, Inc.
MSFT	Microsoft Corporation
MSI	Micro-Star International Corp.
MST	Morning Star Technologies, Inc.
MSystems	M-Systems Flash Disk Pioneers
MTI	MTI Technology Corporation
MTNGATE	MountainGate Data Systems
MXI	Memory Experts International
nac	nac Image Technology Inc.
NAGRA	Nagravision SA - Kudelski Group
NAI	North Atlantic Industries
NAKAMICH	Nakamichi Corporation
NatInst	National Instruments
NatSemi	National Semiconductor Corp.
NCITS	InterNational Committee for Information Technology Standards (INCITS)
NCL	NCL America
NCR	NCR Corporation
Neartek	Neartek, Inc.
NEC	NEC
NETAPP	Network Appliance
NetBSD	The NetBSD Foundation
Netcom	Netcom Storage
NEWISYS	Newisys Data Storage
Newtech	Newtech Co., Ltd.

**Table E.1 — T10 vendor identification list** (part 9 of 13)

<b>ID</b>	<b>Organization</b>
NEXSAN	Nexsan Technologies, Ltd.
NHR	NH Research, Inc.
NISCA	NISCA Inc.
NISHAN	Nishan Systems Inc.
NKK	NKK Corp.
NRC	Nakamichi Research Corporation
NSD	Nippon Systems Development Co.,Ltd.
NSM	NSM Jukebox GmbH
nStor	nStor Technologies, Inc.
NT	Northern Telecom
NUCONNEX	NuConnex
NUSPEED	NuSpeed, Inc.
NVIDIA	NVIDIA Corporation
OAI	Optical Access International
OCE	Oce Graphics
OKI	OKI Electric Industry Co.,Ltd (Japan)
Olidata	Olidata S.p.A.
OMI	Optical Media International
OMNIFI	Rockford Corporation - Omnifi Media
OMNIS	OMNIS Company (FRANCE)
Ophidian	Ophidian Designs
Optelec	Optelec BV
Optiarc	Sony NEC Optiarc Inc.
OPTIMEM	Cipher/Optimem
OPTOTECH	Optotech
ORANGE	Orange Micro, Inc.
ORCA	Orca Technology
OSI	Optical Storage International
OTL	OTL Engineering
OVERLAND	Overland Storage Inc.
pacdigit	Pacific Digital Corp
Packard	Parkard Bell
PARALAN	Paralan Corporation
PASCOsci	Pasco Scientific
PATHLGHT	Pathlight Technology, Inc.
PerStor	Perstor
PERTEC	Pertec Peripherals Corporation
PFTI	Performance Technology Inc.
PFU	PFU Limited
PHILIPS	Philips Electronics
PICO	Packard Instrument Company
Pillar	Pillar Data Systems
PIONEER	Pioneer Electronic Corp.
Pirus	Pirus Networks
PIVOT3	Pivot3, Inc.
PLASMON	Plasmon Data
Pliant	Pliant Technology, Inc.



**Table E.1 — T10 vendor identification list** (part 10 of 13)

<b>ID</b>	<b>Organization</b>
PMCSIERA	PMC-Sierra
POLYTRON	PT. HARTONO ISTANA TEKNOLOGI
PRAIRIE	PrairieTek
PREPRESS	PrePRESS Solutions
PRESOFT	PreSoft Architects
PRESTON	Preston Scientific
PRIAM	Priam
PRIMAGFX	Primagraphics Ltd
PROCOM	Procom Technology
PROMISE	PROMISE TECHNOLOGY, Inc
PROSTOR	ProStor Systems, Inc.
PTI	Peripheral Technology Inc.
PTICO	Pacific Technology International
QIC	Quarter-Inch Cartridge Drive Standards, Inc.
QLogic	QLogic Corporation
Qsan	QSAN Technology, Inc.
QUALSTAR	Qualstar
QUANTEL	Quantel Ltd.
QUANTUM	Quantum Corp.
QUIX	Quix Computerware AG
R-BYTE	R-Byte, Inc.
RACALREC	Racal Recorders
RADITEC	Radikal Technologies Deutschland GmbH
RADSTONE	Radstone Technology
RASVIA	Rasvia Systems, Inc.
rave-mp	Go Video
Readboy	Readboy Ltd Co.
Realm	Realm Systems
Revivio	Revivio, Inc.
RGI	Raster Graphics, Inc.
RHAPSODY	Rhapsody Networks, Inc.
RHS	Racal-Heim Systems GmbH
RICOH	Ricoh
RODIME	Rodime
Royaltek	RoyalTek company Ltd.
RPS	RPS
RTI	Reference Technology
S-D	Sauer-Danfoss
SAMSUNG	Samsung Electronics Co., Ltd.
SAN	Storage Area Networks, Ltd.
Sandial	Sandial Systems, Inc.
SanDisk	SanDisk Corporation
SANKYO	Sankyo Seiki
SANRAD	SANRAD Inc.
SANYO	SANYO Electric Co., Ltd.
SC.Net	StorageConnections.Net
SCIENTEK	SCIENTEK CORP

**Table E.1 — T10 vendor identification list (part 11 of 13)**

<b>ID</b>	<b>Organization</b>
SCInc.	Storage Concepts, Inc.
SCREEN	Dainippon Screen Mfg. Co., Ltd.
SDI	Storage Dimensions, Inc.
SDS	Solid Data Systems
SEAC	SeaChange International, Inc.
SEAGATE	Seagate
SEAGRAND	SEAGRAND In Japan
SEQUOIA	Sequoia Advanced Technologies, Inc.
Shinko	Shinko Electric Co., Ltd.
SIEMENS	Siemens
SigmaTel	SigmaTel, Inc.
SII	Seiko Instruments Inc.
SIMPLE	SimpleTech, Inc. (Obs - now STEC, Inc.)
SLCNSTOR	SiliconStor, Inc.
SLI	Sierra Logic, Inc.
SMS	Scientific Micro Systems/OMTI
SMX	Smartronix, Inc.
SNYSIDE	Sunnyside Computing Inc.
SoftLock	Softlock Digital Security Provider
SONIC	Sonic Solutions
SoniqCas	SoniqCast
SONY	Sony Corporation Japan
SPD	Storage Products Distribution, Inc.
SPECIAL	Special Computing Co.
SPECTRA	Spectra Logic, a Division of Western Automation Labs, Inc.
SPERRY	Sperry (now Unisys Corp.)
STEC	STEC, Inc.
Sterling	Sterling Diagnostic Imaging, Inc.
STK	Storage Technology Corporation
STONEFLY	StoneFly Networks, Inc.
STOR	StorageNetworks, Inc.
STORAPP	StorageApps, Inc.
STORCOMP	Storage Computer Corporation
STORM	Storm Technology, Inc.
StorMagc	StorMagic
Stratus	Stratus Technologies
StrmLgc	StreamLogic Corp.
SUMITOMO	Sumitomo Electric Industries, Ltd.
SUN	Sun Microsystems, Inc.
SUNCORP	SunCorporation
suntx	Suntx System Co., Ltd
SYMANTEC	Symantec Corporation
SYMBIOS	Symbios Logic Inc.
SYNERWAY	Synerway
SyQuest	SyQuest Technology, Inc.
SYSGEN	Sysgen
T-MITTON	Transmitton England

**Table E.1 — T10 vendor identification list** (part 12 of 13)

<b>ID</b>	<b>Organization</b>
T-MOBILE	T-Mobile USA, Inc.
T11	INCITS Technical Committee T11
TALARIS	Talaris Systems, Inc.
TALLGRAS	Tallgrass Technologies
TANDBERG	Tandberg Data A/S
TANDEM	Tandem (now HP)
TANDON	Tandon
TCL	TCL Shenzhen ASIC Micro-electronics Ltd
TDK	TDK Corporation
TEAC	TEAC Japan
TECOLOTE	Tecolote Designs
TEGRA	Tegra Varityper
Tek	Tektronix
TENTIME	Laura Technologies, Inc.
Test	Test new Vendor ID software - delete me!
TGEGROUP	TGE Group Co.,LTD.
TI-DSG	Texas Instruments
TiGi	TiGi Corporation
TILDESGN	Tildesign bv
Tite	Tite Technology Limited
TMS	Texas Memory Systems, Inc.
TMS100	TechnoVas
TOLISGRP	The TOLIS Group
TOSHIBA	Toshiba Japan
TRIPACE	Tripace
TROIKA	Troika Networks, Inc.
TRULY	TRULY Electronics MFG. LTD.
TRUSTED	Trusted Data Corporation
TSSTcorp	Toshiba Samsung Storage Technology Corporation
UDIGITAL	United Digital Limited
UIT	United Infomation Technology
ULTRA	UltraStor Corporation
UNISYS	Unisys
USCORE	Underscore, Inc.
USDC	US Design Corp.
VDS	Victor Data Systems Co., Ltd.
VERBATIM	Verbatim Corporation
Vercet	Vercet LLC
VERITAS	VERITAS Software Corporation
VEXCEL	VEXCEL IMAGING GmbH
VicomSys	Vicom Systems, Inc.
VIDEXINC	Videx, Inc.
VITESSE	Vitesse Semiconductor Corporation
VIXEL	Vixel Corporation
VMAX	VMAX Technologies Corp.
Vobis	Vobis Microcomputer AG
VOLTAIRE	Voltaire Ltd.

**Table E.1 — T10 vendor identification list (part 13 of 13)**

<b>ID</b>	<b>Organization</b>
VRC	Vermont Research Corp.
VRugged	Vanguard Rugged Storage
Waitec	Waitec NV
WangDAT	WangDAT
WANGTEK	Wangtek
Wasabi	Wasabi Systems
WAVECOM	Wavecom
WD	Western Digital Technologies Inc.
WDC	Western Digital Technologies inc.
WDIGTL	Western Digital
WEARNES	Wearnes Technology Corporation
WSC0001	Wisecom, Inc.
X3	InterNational Committee for Information Technology Standards (INCITS)
XEBEC	Xebec Corporation
XENSRC	XenSource, Inc.
Xerox	Xerox Corporation
XIOtech	XIOtech Corporation
XIRANET	Xiranet Communications GmbH
XIV	XIV (now IBM)
XYRATEX	Xyratex
YINHE	NUDT Computer Co.
YIXUN	Yixun Electronic Co.,Ltd.
YOTTA	YottaYotta, Inc.
Zarva	Zarva Digital Technology Co., Ltd.
ZETTA	Zetta Systems, Inc.